ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

"ESTUDIO DE MECANISMOS DE SEGURIDAD EN REDES INALÁMBRICAS DE SENSORES"

Alumno: Patricia Chivite Fernández

Tutor: José Javier Astrain Escola

Pamplona, 15 de Septiembre de 2010

Agradecimientos

Me gustaría mostrar mi agradecimiento a mi tutor, José Javier Astrain Escola, por brindarme la oportunidad de llevar a cabo este proyecto para finalizar una etapa de mis estudios universitarios así como por su apoyo y sus buenos consejos tanto en el ámbito educativo como personales.

Agradecer al tribunal que me preste su tiempo para poder exponer el trabajo realizado cumpliendo así con el último de los objetivos exigidos para acabar este periodo académico.

Asimismo mencionar de una manera especial a Carlos Aristu López por su constante ayuda a lo largo de todo el desarrollo del proyecto ya que sin él creo que no habrían sido posibles lograr muchos de los objetivos.

Por último acordarme de forma general de todas las personas que han hecho posible que este proyecto finalice correctamente tanto por la ayuda prestada a nivel práctico para el desarrollo del proyecto como por su apoyo y consejos personales.

ÍNDICE

1. REDES INALÁMBRICAS……4
1.1 REDES AD-HOC CON SENSORES INALÁMBRICOS…… ...5
2.1.1 Redes inalámbricas…….……...…5
2.1.2 Redes ad-hoc……..……..7
2.1.3 Redes ad-hoc con sensores inalámbricos…….……8
1.2 NUESTRO TRABAJO……..9
1.3 ENTORNO DE TRABAJO: HARDWARE Y SOFTWARE…….……..9
1.3.1 Hardware: Imote2……..……...…….10
1.3.2 Software…….……...…..…12

2. AGREGACIÓN DE DATOS SEGURA EN REDES DE SENSORES INALÁMBRICOS……..……...14
2.1 INTRODUCCIÓN A LA AGREGACIÓN DE DATOS.....…….....15
2.2 LAS EXIGENCIAS DE SEGURIDAD..…….……16
2.3 LA AGREGACIÓN DE DATOS……..……18
2.3.1 Protocolos de agregación de datos jerárquicos…….……19
2.3.2 Protocolos de agregación de datos no jerárquicos……22
2.4 LA AGREGACIÓN SEGURA DE DATOS…….24
2.4.1 La agregación segura de datos que usa datos simples……..……25
2.4.2 Agregación de datos segura que usa datos cifrados…….29
2.5 LA INVESTIGACIÓN ABIERTA Y FUTURAS INVESTIGACIONES…....32

3. DISEÑO PROYECTO……..……34
3.1 ANÁLISIS DEL PROYECTO…….35
3.2 PROBLEMAS EN LA IMPLEMENTACIÓN DEL ALGORITMO……..….35
3.3 GENERAR LAS CLAVES……...…37
3.4 BASE……..……..39
3.5 NODO……..…….39
3.6 DIAGRAMA DE SECUENCIA CON RSA..……..40
3.7 DIAGRAMA DE SECUENCIA CON XTEA ……..……..43
3.7.1 TEA y XTEA……..43
3.7.2 Cambios en el código……..……43

4. RESULTADOS OBTENIDOS……..……..……46
4.1 RESULTADOS OBTENIDOS…….47
4.1.1 Ejecuciones de doscientas interacciones....……47
4.1.2 Ejecuciones de quinientas interacciones…….49
4.1.3 Ejecuciones de mil interacciones……50
4.1.4 Comparativas de los resultados……..51

5. CONCLUSIONES Y MEJORAS……55
5.1 CONCLUSIONES……56
5.2 MEJORAS……57

6. REFERENCIAS……59

1. REDES INALÁMBRICAS

En este capítulo trataremos de explicar las principales características de las redes inalámbricas, del principal problema sobre el que centraremos el trabajo y del hardware y software que hemos elegido para llevar a cabo el proyecto.

1.1 REDES AD-HOC CON SENSORES INALÁMBRICOS

En este apartado trataré de explicar las principales características de una red con sensores inalámbricos.

1.1.1 Redes inalámbricas

El término red inalámbrica *(Wireless network)* se utiliza para designar la conexión de nodos sin necesidad de una conexión física (cables). Ésta se da por medio de ondas electromagnéticas.

Una de sus principales ventajas se encuentra en el coste, ya que se elimina todo el cableado y las conexiones físicas entre nodos; pero también tiene una desventaja considerable, ya que para este tipo de red se debe tener una seguridad mucho más exigente y robusta para evitar intrusiones y escuchas no autorizadas.

En la actualidad, las redes inalámbricas son una de las tecnologías más utilizadas.

1.1.1.1 Categorías

Existen dos categorías en las redes inalámbricas:

- *Larga distancia*: éstas son utilizadas para distancias grandes (kilómetros).
- *Corta distancia*: son utilizadas para distancias más pequeñas (metros).

1.1.1.2 Tipos

Según su cobertura, se pueden clasificar en diferentes tipos:

- *Wireless Personal Area Network (WPAN):* En este tipo de red de cobertura personal, existen tecnologías basadas en HomeRF (estándar para conectar todos los teléfonos móviles de la casa y los ordenadores mediante un aparato central); Bluetooth (protocolo que sigue la especificación IEEE 802.15.1); ZigBee (basado en la especificación IEEE 802.15.4 y utilizado en aplicaciones como la domótica, que requieren comunicaciones seguras con bajas tasas de transmisión de datos y bajo consumo); RFID (siglas de *Radio Frequency IDentification*, en castellano identificación por radiofrecuencia, cuyo propósito fundamental es transmitir la identidad de un objeto y sus datos (similar a un número de serie único) mediante ondas de radio.

- *Wireless Local Area Network (WLAN):* En las redes de área local podemos encontrar tecnologías inalámbricas basadas en HiperLAN (del inglés, *High Performance Radio LAN*), un estándar del grupo ETSI, o tecnologías basadas en Wi-Fi, que siguen el estándar IEEE 802.11 con diferentes variantes.

- *Wireless Metropolitan Area Network:* Para redes de área metropolitana se encuentran tecnologías basadas en WiMAX (*Worldwide Interoperability for*

Microwave Access, es decir, Interoperabilidad Mundial para Acceso con Microondas), un estándar de comunicación inalámbrica basado en la norma IEEE 802.16. WiMAX es un protocolo parecido a Wi-Fi, pero con más cobertura y ancho de banda. También podemos encontrar otros sistemas de comunicación como LMDS (*Local Multipoint Distribution Service*).

- *Wireless Wide Area Network (WAN):* En estas redes encontramos tecnologías como UMTS (*Universal Mobile Telecommunications System*), utilizada con los teléfonos móviles de tercera generación (3G) y sucesora de la tecnología GSM (para móviles 2G), o también la tecnología digital para móviles GPRS, *General Packet Radio Service* (2,5G).

1.1.1.3. Características

Según el rango de frecuencias utilizado para transmitir, el medio de transmisión pueden ser las ondas de radio, las microondas terrestres o por satélite, y los infrarrojos, por ejemplo. Dependiendo del medio, la red inalámbrica tendrá unas características u otras:

- *Ondas de radio:* las ondas electromagnéticas no necesitan de un medio físico de transporte por lo que pueden propagarse tanto a través del aire como del espacio vacío. La transmisión no es sensible a las atenuaciones producidas por la lluvia ya que se opera en frecuencias no demasiado elevadas. En este rango se encuentran las bandas desde la ELF que va de 3 a 30 Hz, hasta la banda UHF que va de los 300 a los 3000 MHz, es decir, comprende el espectro radioeléctrico de 30 – 3.000.000 Hz.

- *Microondas terrestres:* se utilizan antenas parabólicas de diferentes diámetros según el uso que se les vaya a dar. Tienen una cobertura de kilómetros, pero con el inconveniente de que el emisor y el receptor deben estar perfectamente alineados. Por eso, se acostumbran a utilizar en enlaces punto a punto en distancias cortas. En este caso, la atenuación producida por la lluvia es más importante ya que se opera a una frecuencia más elevada. Las microondas comprenden las frecuencias desde 1 hasta 300 GHz.

- *Microondas por satélite:* se hacen enlaces entre dos o más estaciones terrestres que se denominan estaciones base. El satélite recibe la señal (denominada señal ascendente) en una banda de frecuencia, la amplifica y la retransmite en otra banda (señal descendente). Cada satélite opera en unas bandas concretas. Las comunicaciones entre ellos pueden convivir en el espacio y tiempo mientras que las frecuencias no se mezclan.

- *Infrarrojos:* se enlazan transmisores y receptores que modulan la luz infrarroja no coherente. Para trabajar correctamente deben estar alineados directamente o de manera que exista una reflexión muy pequeña. No pueden atravesar las paredes.

1.1.1.4. Aplicaciones

Las bandas más importantes con aplicaciones inalámbricas, del rango de frecuencias que abarcan las ondas de radio, son la VLF (comunicaciones en navegación y submarinos), LF (radio AM de onda larga), MF (radio AM de onda media), HF (radio AM de onda corta), VHF (radio FM y TV) y UHF (TV).

- *Mediante las microondas terrestres*, existen diferentes aplicaciones basadas en protocolos como Bluetooth o ZigBee para interconectar ordenadores portátiles, PDAs, teléfonos u otros aparatos. También se utilizan las microondas para comunicaciones con radares (detección de velocidad u otras características de objetos remotos) y para la televisión digital terrestre.

- *Las microondas por satélite* se usan para la difusión de televisión por satélite, transmisión telefónica a larga distancia y en redes privadas, por ejemplo.

- *Los infrarrojos* tienen aplicaciones como la comunicación a corta distancia de los ordenadores con sus periféricos. También se utilizan para mandos a distancia, ya que así no interfieren con otras señales electromagnéticas, por ejemplo la señal de televisión. Uno de los estándares más usados en estas comunicaciones es el IrDA (*Infrared Data Association*). Otros usos que tienen los infrarrojos son técnicas como la termografía, la cual permite determinar la temperatura de objetos a distancia.

1.1.1.5. Topologías

Existen dos topologías de redes inalámbricas:

- *Ad-Hoc:* Una red ad hoc es una red descentralizada. La red es ad hoc porque cada nodo está preparado para reenviar datos a los demás y la decisión sobre qué nodos reenvían los datos se toma de forma dinámica en función de la conectividad de la red. Esto contrasta con las redes tradicionales en las que los *routers* llevan a cabo esa función. También difiere de las redes inalámbricas convencionales en las que un nodo especial, llamado punto de acceso, gestiona las comunicaciones con el resto de nodos.

- *De infraestructura*: Se utiliza un dispositivo como punto de acceso, que centraliza la información. Los dispositivos le envían los paquetes de información y éste se ocupa de enviar cada uno de ellos al dispositivo destinatario correspondiente.

1.1.2. Redes ad-hoc

Una red ad hoc es una red descentralizada. Anteriormente ya se han descrito sus principales características. En este tipo de redes pueden convivir los dispositivos inalámbricos con parte cableada.

La naturaleza descentralizada de las redes ad hoc, hace de ellas las más adecuadas en aquellas situaciones en las que no puede confiarse en un nodo central y mejora su escalabilidad comparada con las redes inalámbricas tradicionales, desde el punto de vista teórico y práctico.

Las redes ah hoc son también útiles en situaciones de emergencia, como desastres naturales o conflictos bélicos, al requerir muy poca configuración y permitir un despliegue rápido. El protocolo de encaminamiento dinámico permite que entren en funcionamiento en un tiempo muy reducido.

Por su aplicación pueden clasificarse como:

- *Redes móviles ad hoc (MANETs):* es un sistema autónomo de nodos móviles conectados de forma inalámbrica. Cada nodo no sólo opera como un fin de sistema, también como un *router* para retransmitir los paquetes. Las redes móviles ad hoc no requieren una infraestructura fija.

- *Redes inalámbricas mesh*: son aquellas redes en las que se mezclan las dos topologías de las redes inalámbricas, la topología Ad-hoc y la topología infraestructura. Básicamente son redes con topología de infraestructura pero que permiten unirse a la red a dispositivos que, a pesar de estar fuera del rango de cobertura de los puntos de acceso, están dentro del rango de cobertura de alguna tarjeta de red (TR) que directamente o indirectamente está dentro del rango de cobertura de un punto de acceso (PA).

- *Redes de sensores*: Las redes de sensores están formadas por un grupo de sensores con ciertas capacidades sensitivas y de comunicación inalámbrica los cuales permiten formar redes *ad hoc* sin infraestructura física preestablecida ni administración central.

1.1.3. Redes ad-hoc con sensores inalámbricos

Una red de sensores se define como un tipo particular de red inalámbrica, formada por un conjunto de dispositivos o sensores con unas limitaciones en cuanto a memoria, ancho de banda, velocidad de procesado y reservas de energía mucho más restrictivas que en el caso de los nodos de las redes inalámbricas tradicionales.

Además, estas redes se caracterizan por estar formadas por un gran número de dispositivos (del orden de cientos o miles) de reducidas dimensiones y bajo coste, siendo un factor clave en el diseño de las mismas una implantación rentable en términos económicos. Debido a su naturaleza ad-hoc, a la mínima intervención humana sobre ellas, así como a su alto grado de escalabilidad, las redes de sensores pueden presentar verdaderas dificultades para crear un sistema de identificación y autoconfiguración de las mismas.

Otro aspecto particular de este tipo de redes es el amplio abanico de protocolos de encaminamiento que pueden soportar en función del tipo de aplicación requerida. Así, además de los tradicionales protocolos ‘adress centric’, en los que se direccionan los nodos, las redes de sensores pueden ejecutar otros protocolos mucho más eficientes

para la recolección de datos de un entorno determinado, como pueden ser los 'data centric'. En los primeros protocolos mencionados en muy importante el nodo que manda la información, hay que tener presente la identidad de dicho nodo mientras que en el segundo lo más importante es el mensaje que se manda siendo de menor importancia la identidad de dicho nodo.

1.2 NUESTRO TRABAJO

Tras haber explicado las características de las redes de sensores inalámbricas cabe destacar uno de sus principales problemas. Estas redes están formadas por dispositivos de poca potencia computacional y con memoria limitada. Una de las limitaciones que más condicionan el diseño de estas redes es la potencia de la batería. Para reducir el consumo se proponen varios mecanismos como la planificación de radio, controlar la eliminación de paquetes, el control de la topología, y el más importante y en el que nos centraremos, la agregación de datos. Esta técnica y los estudios llevados a cabo sobre ella hasta el momento serán explicados en profundidad en el próximo capítulo.

El trabajo en este proyecto va a consistir en el estudio de los mecanismos para proveer seguridad en redes de sensores inalámbricas. En concreto, sobre redes de sensores *Crossbow imote2*® [3].

Los principales sistemas de seguridad manejan diferentes mecanismos de cifrado y autentificación, así como mecanismos de detección de intrusión. En este marco de trabajo, la gestión de claves de cifrado y firma digital, así como los algoritmos criptográficos, los protocolos de autenticación y los mecanismos de detección de intrusos o clones son aspectos muy a tener en cuenta en este tipo de sistemas.

Con este proyecto se pretende desarrollar un prototipo de sistema en el que se pueda autentificar a los diferentes sensores de una red WSN (*wireless sensor network*), se permita la comunicación segura (autentificada y cifrada) entre nodos de una misma subred, y se pueda detectar intrusiones en la red.

Haremos un estudio comparativo de dicho algoritmo cuando usa diferentes mecanismos de encriptación de los datos. El algoritmo en cuestión se estudiará en un capítulo próximo y mediante los resultados obtenidos trataremos de comprobar si nos proporciona los objetivos fundamentales: la seguridad y la minimización del consumo de energía por parte de los sensores para aumentar el tiempo de vida de la red.

1.3 ENTORNO DE TRABAJO: HARDWARE Y SOFTWARE

En este apartado explicaremos el hardware y el software elegido para llevar a cabo nuestro proyecto.

1.3.1 Hardware: Imote2

El término hardware corresponde a todas las partes físicas de un ordenador: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado. Por otro lado encontramos el término software que hace referencia al equipamiento lógico o soporte lógico necesarios en un ordenador para hacer posible la realización de tareas específicas.

Para llevar a cabo nuestro proyecto hemos elegido unos sensores de la compañía Crossbow [3]. Dichos sensores reciben el nombre de Imote2, de los cuales podemos detallar sus características principales:

- Intel PXA271 XScale ® Procesador en 13 - 416MHz
- Radio de Intel MMX DSP Coprocesador
- 256 kB SRAM, 32 MB FLASH, 32MB
- Radio integrada 802.15.4
- Antena integrado 2.4GHz, Conector Opcional Externo SMA
- Indicador de Estado Multicolor LED
- USB Cliente y adaptadores de host
- Conjunto de entrada/salida: 3xUART, 2xSPI, I2C, SDIO, GPIOs
- Entrada/salida específicas de aplicación: I2S, AC97, Interfaz de cámara, JTAG
- Tamaño compacto: 36mm x 48mm x 9mm

FIGURA 1**:** Sensor Imote2

El Imote2 es un sensor inalámbrico avanzado. Se construye sobre el procesador Intel PXA271 XScale e integra una interfaz radio 802.15.4 compatible. El diseño es

modular y apilable (es decir, se pueden conectar varios sensores uno sobre otro) con interfaz de conectores para tarjetas de expansión tanto en la parte superior como inferior. Los conectores superiores proporcionan un conjunto estándar de señales de entrada/salida. Los conectores inferiores proporcionan interfaces adicionales de alta velocidad para la entrada/salida específica de la aplicación.

1.3.1.1 Procesador

El Imote2 contiene el procesador Intel PXA27. Este procesador puede funcionar con un voltaje bajo (0.85V) y a baja frecuencia (13MHz), de ahí que sólo permita el funcionamiento de bajo coste operacional. La frecuencia se puede escalar desde 13MHz a 416MHz con lo que conseguimos el ahorro de la energía. El procesador tiene pocos modos de energía. El PXA271 es un módulo multi-chip que incluye tres chips en un solo paquete, la CPU con 256 KB de SRAM, la SDRAM de 32 MB y 32 MB de memoria FLASH. Esto integra muchas opciones de entrada/salida que incluyen características I2C, 2 Puertos serie Sincrónicos (SPI) uno de los cuales está dedicado a la radio, 3 UARTs de alta velocidad, GPIOs, SDIO, un puerto rápido de infrarrojos, PWM, un Interfaz de Cámara y un bus de alta velocidad. El procesador también es compatible con numerosos temporizadores así como con un reloj en tiempo real. El PXA271 incluye una conexión inalámbrica MMX para acelerar operaciones. Esto añade nuevos mecanismos de comunicación, apoyo a la alineación y operaciones de vídeo y compatibilidad con Intel MMX e instrucciones de número entero SSE.

1.2.1.2 Radio y antena

El Imote2 usa el transmisor-receptor de radio CC2420 IEEE 802.15.4. El CC2420 es compatible con una velocidad de transferencia de datos de 250kb/s con 16 canales en la banda de 2.4GHz. La plataforma integra una antena superficial que proporciona un rango nominal de unos 30 metros. Para mayor alcance se pueden soldar conectores SMA directamente a la tarjeta para conectarse a una antena externa.

1.2.1.3 Fuente de alimentación

El Imote2 puede ser alimentado de varias formas:

- *Batería primaria*: Esto se consigue juntando una batería al Imote2 mediante los conectores básicos o avanzados.

- *Batería recargable*: Esto requiere una batería configurada de manera especial y conectada mediante los conectores básicos o los avanzados. El Imote2 tiene un cargador empotrado para Li-ion o pilas Li-Poly.

- *USB:* El Imote2 puede ser alimentado a través de un conector USB. Este modo también puede ser usado para cargar una batería conectada.

- *Almohadillas de Batería*: Una batería primaria u otra fuente de energía pueden ser conectadas a través de una serie de pastillas de soldadura a la tarjeta Imote2.

1.2.1.4 Batería

La batería universal de Imote2 está diseñada para alimentar el Imote2 con tres células primarias AAA. También pueden ser usadas células recargables como AAA NiMH.

Un interruptor mecánico a un lado proporciona la capacidad de apagar y encender la batería.

FIGURA 2**:** Batería de un sensor Imote2.

1.3.2 Software

1.3.2.1 .NET

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en la transparencia de redes, con independencia de la plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

La plataforma .NET de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

Los principales componentes del marco de trabajo son:

- *El conjunto de lenguajes de programación*

- *La Biblioteca de Clases Base o BCL*

- *El Entorno Común de Ejecución para Lenguajes o CLR*

Debido a la publicación de la norma para la infraestructura común de lenguajes (*CLI* por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualesquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Algunos de los lenguajes desarrollados para el marco de trabajo .NET son: C#, Visual Basic, Delphi (Object Pascal), C++, J#, Perl, Python, Fortran, Prolog, Cobol y PowerBuilder.

1.3.2.2 C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA [1] e ISO [2].

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguajes (entre ellos Delphi).

La creación del nombre del lenguaje, C#, proviene de dibujar dos signos positivos encima de los dos signos positivos de "C++", queriendo dar una imagen de salto evolutivo del mismo modo que ocurrió con el paso de C a C++.

C#, como parte de la plataforma .NET, está normalizado por ECMA desde diciembre de 2001 (C# Language Specification "Especificación del lenguaje C#"). El 7 de noviembre de 2005 salió la versión 2.0 del lenguaje que incluía mejoras tales como tipos genéricos, métodos anónimos, iteradores, tipos parciales y tipos anulables. El 19 de noviembre de 2007 salió la versión 3.0 de C# destacando entre las mejoras los tipos implícitos, tipos anónimos y LINQ (*Language Integrated Query* -consulta integrada en el lenguaje).

Aunque C# forma parte de la plataforma .NET, ésta es una interfaz de programación de aplicaciones (**API**); mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco de DotGNU - Mono que genera programas para distintas plataformas como Win32, UNIX y Linux.

2. AGREGACIÓN DE DATOS SEGURA EN REDES DE SENSORES INALÁMBRICOS

En este capítulo centraremos el trabajo en una de las principales técnicas que se usan en las redes de sensores inalámbricas, la agregación de datos y detallaremos los estudios llevados a cabo sobre este tema hasta el momento.

2.1 INTRODUCCIÓN A LA AGREGACIÓN DE DATOS

Como ya he mencionado, las redes de sensores inalámbricos están compuestas por cientos o miles de dispositivos de poca potencia computacional y con memoria limitada. Estas redes ofrecen soluciones potencialmente económicas a una serie de problemas tanto en usos militares como en usos cotidianos, incluyendo, por ejemplo, la vigilancia del campo de batalla, el rastreo del objetivo, usos ambientales, la supervisión de asistencia médica, la detección de fuego o gases y la regulación de tráfico. Debido al bajo coste de los requisitos de implementación, los nodos de la red tienen el hardware simple y múltiples limitaciones de recursos. Entre estas limitaciones, "la potencia de la batería" es el factor más restrictivo en el diseño de las redes de sensores inalámbricos. Por lo tanto, para reducir el consumo de electricidad en las redes de sensores inalámbricos, se proponen varios mecanismos como la planificación del uso el interfaz de radio, controlar la eliminación de paquetes, el control de la topología, y el más importante, la agregación de datos. Los protocolos de agregación de datos tienen como objetivo combinar y resumir los paquetes de datos de varios nodos de modo que se reduzca la cantidad de información transmitida. En la Figura 3 se presenta un ejemplo de un esquema de agregación de datos, donde un grupo de nodos recoge la información de una región. En ocasiones, en lugar de enviar cada uno de los nodos sus datos a la estación base, existe un nodo llamado agregador de datos que recoge la información de sus nodos vecinos y envía todos los datos agregados a la estación base sobre un camino de multisalto. La agregación de datos reduce el número de transmisiones de información, así se mejoran la amplitud de banda y la utilización de la energía en la red. En redes de sensores inalámbricos, las ventajas de la agregación de datos aumentan si los nodos intermedios realizan la agregación de datos incrementalmente cuando éstos están siendo enviados a la estación base. Sin embargo, mientras esta operación de agregación de datos continúa y mejora el ancho de banda y el consumo energético, esto puede afectar negativamente a otras métricas de funcionamiento como el retraso, la exactitud, la tolerancia de fallos y la seguridad. Como la mayoría de los usos de redes de sensores inalámbricos requiere un cierto nivel de seguridad, no es posible sacrificar la seguridad para la agregación de datos. Además, hay un conflicto fuerte entre la seguridad y los protocolos de agregación de datos. Los protocolos de seguridad requieren que los nodos cifren y autentifiquen los datos antes de su transmisión y prefieren que los datos sean descifrados por la estación base. Por otra parte, los protocolos de agregación de datos prefieren datos simples para poner en práctica la agregación de datos en cada nodo intermedio de modo que la eficacia de la energía sea maximizada.

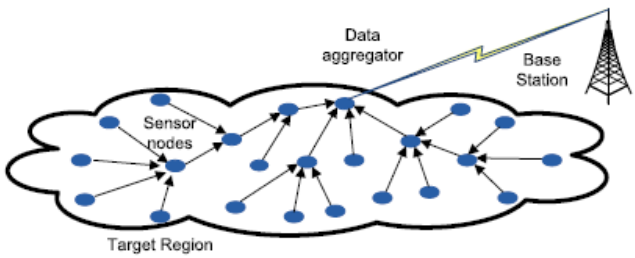


FIGURA 3: Agregación de datos en una red de sensores inalámbricos

Además, la agregación de datos causa alteraciones en los datos y por lo tanto la tarea de proporcionar la fuente y la autenticación de los datos junto con la agregación de datos es un reto. Debido a estos objetivos contrapuestos, la agregación de datos y los protocolos de seguridad deben ser diseñados conjuntamente de modo que la agregación de datos pueda ser realizada sin sacrificar la seguridad. La necesidad de poner en práctica la agregación de datos y la seguridad de éstos juntos ha conducido a muchos investigadores a trabajar sobre el problema de la agregación de datos segura. Comparado al problema de agregación de datos general que es un tema ampliamente investigado, el problema de la agregación confiable de datos todavía puede proporcionar muchas investigaciones interesantes.

2.2 LAS EXIGENCIAS DE SEGURIDAD

Debido a los ambientes hostiles en los que trabajan y a las propiedades únicas de las redes de sensores inalámbricos, es una ardua tarea proteger la información sensible transmitida por dichas redes. Además, las redes de sensores inalámbricos tienen problemas de seguridad que las redes tradicionales no afrontan. Por lo tanto, la seguridad es una cuestión importante y hay muchas consideraciones de seguridad que deberían ser investigadas. En esta sección, presentamos las exigencias de seguridad esenciales que se plantean en un ambiente de redes de sensores inalámbricos y se explican cómo estas exigencias se relacionan con el proceso de agregación de datos. La Figura 4 ilustra la interacción entre exigencias de seguridad de las redes de sensores inalámbricos y el proceso de agregación de datos.

- *La confidencialidad de datos:* En las redes de sensores inalámbricos, se garantiza la confidencialidad de los datos haciendo que dicha información nunca sea revelada a partes no autorizadas lo cual es una de las cuestiones más importantes en aplicaciones de misiones críticas. Además, en muchos usos, los nodos transmiten datos sumamente sensibles, por ejemplo, claves secretas; y por lo tanto es muy importante construir canales seguros entre los nodos. La información pública de cada sensor, como la identidad y las claves públicas, también debería ser cifrada en cierta medida para protegerse contra ataques de análisis de tráfico. Además, la información de enrutamiento también debe permanecer confidencial ya que en ciertos casos como es el de los nodos malévolos pueden usar esta información para degradar el funcionamiento de la red. El método estándar para mantener en secreto los datos sensibles es cifrar los datos con una clave secreta que sólo posean los receptores. Sin embargo, los protocolos de agregación de datos por lo general no pueden agregar datos cifrados. Por lo tanto, tales protocolos de agregación de datos deben descifrar los datos para realizar la agregación de datos y cifrar los datos agregados antes de la transmisión de éstos. Este descifrado/cifrado no sólo causa el retraso y el consumo de energía, sino que también previene la confidencialidad de datos de extremo a extremo.

- *La integridad de los datos y la frescura:* Aunque la confidencialidad de datos garantice que sólo las partes elegidas obtengan los datos codificados, esto no impide que los datos sean alterados. La integridad de los datos

garantiza que un mensaje siendo transferido nunca es corrompido. Un nodo malévolo puede corromper mensajes para impedir a la red funcionar correctamente. De hecho, debido a canales de comunicación no fiables, los datos pueden ser alterados sin la presencia de un intruso. Así, los códigos de autenticación de mensajes o los códigos cíclicos se emplean para garantizar la integridad de los datos. La agregación de datos causa las alteraciones de datos; por lo tanto, no es posible tener la comprobación de integridad de extremo a extremo cuando se emplea la agregación de datos. Además, si los datos agregados están comprometidos, esto puede corromper datos durante la agregación de datos y la estación base no tiene ningún modo de comprobar la integridad de estos datos agregados. Los nodos comprometidos son capaces de escuchar mensajes transmitidos y volverlos a enviar más tarde para interrumpir los resultados de agregación de datos. La frescura de los datos protege a los esquemas de agregación de datos contra ataques de repetición asegurando que los datos transmitidos son recientes.

- *La autenticación de la fuente*: Las redes de sensores inalámbricos usan un medio inalámbrico compartido, los nodos necesitan mecanismos de autenticación para descubrir paquetes maliciosamente inyectados o falsos. La autenticación de la fuente permite a un nodo de sensor asegurar la identidad del nodo con el que se comunica. Sin la autenticación de la fuente, un adversario podría conseguir el acceso no autorizado al recurso y a la información sensible interfiriendo con la operación de otros nodos. Además, un nodo comprometido puede enviar datos bajo varias identidades falsas de modo que la integridad de los datos agregados sea corrompida. Se llama a la falsificación de múltiples identidades de nodo el ataque de Sybil y esto plantea una amenaza significativa para los protocolos de agregación de datos. Si sólo se comunican dos nodos, pueden proporcionar la autenticación mediante la criptografía de una clave simétrica. El remitente y el receptor comparten una llave secreta para calcular el código de autenticación de mensaje para todos los datos transmitidos. Sin embargo, los datos agregados pueden tener que difundir la autenticación de transmisión que requiere técnicas más complejas.

- *La disponibilidad*: La disponibilidad garantiza la supervivencia de los servicios de la red contra los ataques de negación de servicio. Un ataque de este tipo puede ser lanzado en cualquier capa de un sensor inalámbrico y puede incapacitar el nodo permanentemente. Además, la comunicación excesiva o el cómputo puede agotar la carga de batería de un nodo. Las consecuencias de la pérdida de disponibilidad pueden ser catastróficas. Por ejemplo, en un uso de vigilancia de campo de batalla, si no pueden proporcionar la disponibilidad de algunos nodos, esto puede conducir a una invasión enemiga. Así, en redes de sensores inalámbricos, los intrusos lanzan ataques de este tipo con el objetivo de evitar que los agregadores de datos puedan realizar su tarea de modo que alguna parte de la red ocasione pérdidas en la disponibilidad.

FIGURA 4: La interacción entre una red de sensores inalámbricos y la seguridad y el proceso de agregación de datos.

2.3 LA AGREGACIÓN DE DATOS

En una red de sensores inalámbrica típica, un elevado número de nodos recoge la información específica de la aplicación y ésta es transferida a una estación base central donde es procesada, analizada y usada por la aplicación. El objetivo principal de la agregación de datos es aumentar la vida de la red reduciendo el consumo de recursos de los nodos (como la energía de la batería y el ancho de banda). Aumentando el tiempo de vida de la red, los protocolos de agregación de datos pueden degradar la calidad de algunas métricas como la exactitud de los datos, la latencia, la tolerancia de fallos o la seguridad. Por lo tanto, el diseño de un protocolo de agregación de datos eficiente es una difícil tarea porque el diseñador del protocolo debe equilibrarlo entre la eficacia de la energía, la exactitud de los datos, la latencia, la tolerancia de fallos y la seguridad. Para alcanzar este equilibrio, las técnicas de agregación de datos están relacionadas con el modo en el que se enrutan los paquetes a través de la red. La arquitectura de la red de sensores juega un papel vital en el funcionamiento de los protocolos de agregación de datos. Hay varios protocolos que permiten el envío y la agregación de paquetes de datos simultáneamente. Estos protocolos pueden ser clasificados en dos categorías: protocolos de agregación de datos jerárquicos [4-9] y protocolos de agregación de datos no jerárquicos [10-13]. Los anteriores trabajos sobre la agregación de datos se centraron en la mejora de los algoritmos de enrutamiento existentes a fin de hacer la agregación de datos posible. Por consiguiente, se han propuesto muchos protocolos de agregación de datos basados en una estructura jerárquica y en sus caminos más cortos. Para reducir la latencia debido a la agregación de datos jerárquica, los trabajos más recientes [14-35] sobre la agregación de datos tiende a grupos de nodos con una estructura no jerárquica de modo que los datos sean agregados en cada grupo para mejorar la eficacia.

2.3.1 Protocolos de agregación de datos jerárquicos

El modo más simple de alcanzar la agregación de datos distribuida es determinar algunos nodos agregadores de datos en la red y asegurar que los caminos que seguirán los datos en la red incluyan estos nodos agregadores. Estas técnicas de agregación de datos han sido ampliamente estudiadas. La cuestión principal de los protocolos de agregación de datos jerárquicos es la construcción de un árbol de agregación de datos eficiente con respecto a la energía. La Figura 5 ilustra un ejemplo de agregación de datos jerárquico. El Árbol Ávido Incremental (GIT) [38] es un protocolo de envío céntrico de datos que permite la agregación de datos basada en la Difusión Dirigida [5]. GIT es comparado con otros dos esquemas de envíos céntricos de datos, Centrarse en la Fuente Más cercana (CNS) y el Árbol de Camino Más corto (SPT). Los resultados de simulación muestran que GIT es el que mejores resultados obtiene en términos de número medio de transmisiones. Proponen otro protocolo SPT de agregación de datos que promueve la energía del nodo padre. En este protocolo, la selección paternal está basada en la distancia de los nodos a la estación base y su nivel de energía residual. Hay también protocolos de agregación de datos que consideran la teoría de la información como la métrica de enrutamiento. Sin embargo, este protocolo no es viable ya que depende del conocimiento global de la entropía de la información de cada nodo así como de la entropía conjunta de cada par de nodos.

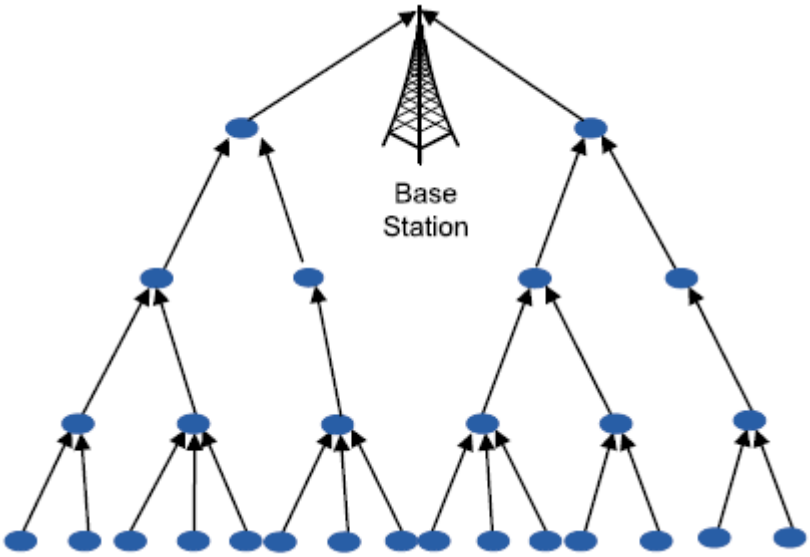


FIGURA 5: Árbol de agregación de datos.

Madden et al. [4] propusieron un marco de agregación *data centric* llamado Servicio Diminuto de agregación (TAG), que está basado en el enrutamiento del camino más corto del árbol. TAG fue diseñado expresamente para supervisar usos y permite una lista ajustable para los nodos. Para alcanzar esto, los nodos “padres” dejan a sus "hijos" saber el tiempo de espera para la transmisión. También, los nodos que actúan como padres esconden los datos de sus hijos para impedir la pérdida de datos. TAG realiza la agregación de datos en dos fases. En la primera fase, denominada de distribución, las preguntas de las estaciones base son difundidas al resto de nodos. En la segunda fase, llamada de recolección, las lecturas de los sensores se enrutan en el árbol de agregación. Durante la fase de distribución, un mensaje es difundido por la estación base que requiere nodos para organizar un árbol de enrutamiento de modo que la estación base

pueda enviar sus preguntas. Cada mensaje tiene un campo que especifica el nivel o la distancia desde la raíz al nodo emisor (el nivel de la raíz es igual al cero). Cuando un nodo que no pertenece a ningún nivel recibe este mensaje, éste pone su propio nivel incrementando el nivel en uno y asigna al remitente como su padre. Este proceso sigue hasta que todos los nodos en la red se unan al árbol y tengan un padre. Esta mensajería repetida ayuda a mantener la estructura de árbol actualizada. Una vez que se forma el árbol, entonces la estación base pregunta a la red a través del árbol de agregación. Los nodos usan a sus padres para dar respuesta a las preguntas de la estación base. TAG emplea SQL como el lenguaje de consulta a la red. Cada consulta especifica la cantidad que debe ser recogida, la función de agregación y los nodos que tienen que realizar la recolección de datos.

La difusión dirigida [5] es un protocolo centrado en los datos (*data-centric*) que se lleva a cabo en tres fases: diseminación del interés, sistema de gradiente, y refuerzo de camino y expedición. En la primera fase, la estación base propaga un mensaje de interés (la diseminación de interés) que describe el tipo de datos que tienen que ser recogidos y el la manera en que se llevará a cabo la recolección. Sobre la recepción de este mensaje, cada nodo difunde de nuevo el mensaje a sus vecinos. Los nodos también preparan los gradientes de interés que son básicamente los vectores que contienen el siguiente salto que tiene que ser usado para propagar el resultado de la pregunta a la estación base (el sistema de gradiente). Para cada tipo de datos puede establecerse un gradiente diferente. En la Figura 6 se presenta un ejemplo ilustrativo de protocolo de difusión dirigido. La estación base actualiza periódicamente el árbol de datos. Sin embargo, esto es una operación costosa y esto puede disminuir el beneficio de la agregación de datos si la red tiene una topología dinámica. Después surgió una versión mejorada de este algoritmo, la llamada Difusión Dirigida Mejorada (EDD) [6], que integra la difusión dirigida con una arquitectura no jerárquica de modo que la eficacia de las interacciones locales durante la fase de gradiente aumente. En PEGASIS [7] se propone el uso eficiente de la energía en los sistemas de recolección de información del sensor que organiza los nodos en una cadena. Cada cadena de agregación de datos tiene un líder que es el responsable de enviar los datos agregados a la estación base. Para distribuir uniformemente el gasto de energía en la red, los nodos se turnan para actuar como líderes de la cadena. La formación de la cadena puede desencadenarse de una manera centralizada por la estación base o de una manera descentralizada usando un algoritmo voraz en cada nodo. Ambos accesos requieren el conocimiento global de la red. El proceso de creación de la cadena comienza del nodo que está más lejos de la estación base y sigue hacia la estación base. Cuando un nodo muere, la cadena se reconstruye. En una cadena de nodos, cada nodo recibe datos de un vecino y lo agrega con su propia lectura generando un único paquete. Este proceso se repite a lo largo de la cadena y el líder agrega sus propios datos al paquete y lo envía a la estación base directamente. Así, el líder en cada ronda de comunicación estará en una posición arbitraria sobre la cadena, lo cual es importante para que la red no fracase. Las dos desventajas principales de PEGASIS son por un lado que requiere que cada nodo tenga una vista completa de la topología de la red de modo que las cadenas se puedan formar correctamente. Además, todos los nodos deben ser capaces de transmitir directamente a la estación base. Por otro lado, si las distancias entre los nodos en una cadena son demasiado grandes, entonces el gasto de energía de los nodos puede ser considerablemente alto.

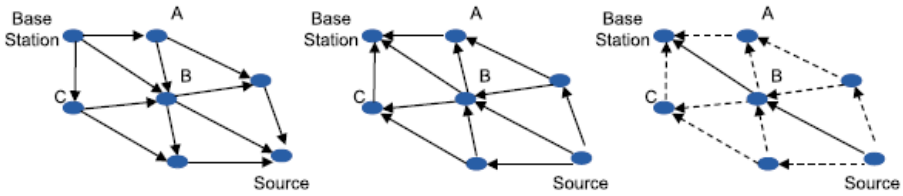


FIGURA 6: Ejemplo ilustrativo de difusión dirigida.

En [8] se propone un protocolo de construcción jerárquica de agregación de datos que sólo se basa en el conocimiento local de la topología de la red. El protocolo propuesto, llamado EADAT, está basado en una heurística conocida basada en la energía. La estación base es la raíz del árbol de agregación de ahí que el árbol se inicia difundiendo un mensaje de control que tiene los cinco campos siguientes: ID, padre, batería, estado, y hopcount (el número de saltos que lleva a cabo un paquete entre el nodo fuente y el destino). Este mensaje se envía entre los nodos hasta que cada nodo difunda el mensaje una vez y el resultado es un árbol de agregación situado en la estación base. Al tener en cuenta el nivel de energía de los nodos, el algoritmo da la posibilidad a los nodos con potencia más alta de convertirse en un nodo hoja. Por lo tanto, el reenvío de los datos es una tarea que se lleva a cabo por los nodos que tienen altos niveles de energía. Los resultados de la simulación muestran que EADAT prolonga la vida de la red y ahorra más energía en comparación con el método de transición sin agregación. También se observa que el nivel de energía medio de los nodos disminuye mucho más despacio comparado con el método sin la agregación de datos.

Hay muchas soluciones adicionales para el problema de construir árboles de agregación de datos en redes de sensores inalámbricas de forma eficiente. Un enfoque diferente, llamado DBMAC [9], integra el enrutamiento y propone protocolos MAC para realizar la agregación de datos. El objetivo principal del esquema propuesto por DBMAC debe reducir al mínimo tanto la latencia como aumentar la eficacia de la energía aprovechando mecanismos de agregación de datos.

2.3.2 Protocolos de agregación de datos no jerárquicos

En los protocolos de agregación de datos no jerárquicos, los nodos son subdivididos en *clusters*. En cada *cluster* se elige un nodo para agregar datos en la zona y transmitir el resultado de la agregación a la estación base. Las *cluster heads* pueden comunicarse con el receptor directamente mediante radiofrecuencia. Sin embargo, esto es bastante ineficaz con respecto a la energía de los nodos. Así, *las cluster heads* por lo general forman una estructura de árbol para transmitir los datos agregados por otras *cluster heads* lo cual causa un ahorro de energía significativo. La Figura 7 presenta un ejemplo de agregación de datos no jerárquica. Recientemente, se han propuesto varios protocolos de agregación de datos de este tipo [10-13].

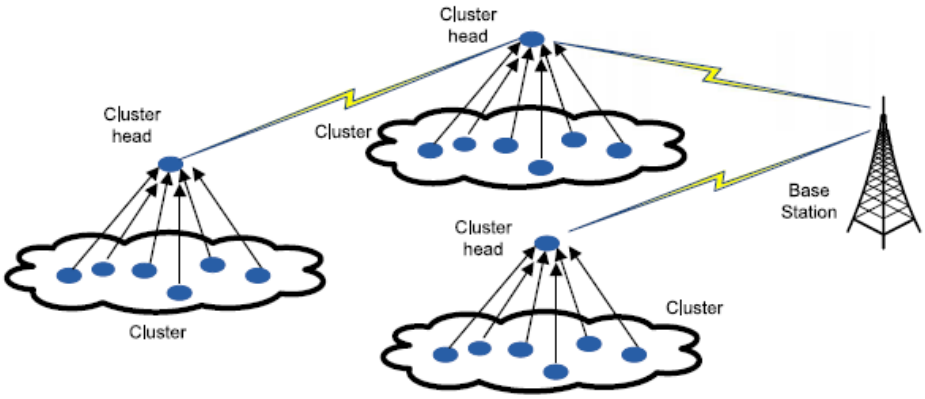


FIGURA 7: Agregación de datos a base de cluster

LEACH [10] aprovecha la ordenación aleatoria de los nodos para distribuir uniformemente el gasto de energía entre estos. LEACH es un enfoque donde las *cluster heads* actúan como puntos de agregación de datos. El protocolo consiste en dos fases. En la primera fase, se forman los *clusters* de la red. Después, en la segunda fase, las *cluster heads* agregan y transmiten los datos a la estación base. El proceso de elección de la *cluster head* del LEACH está basado en un enfoque distribuido probabilístico tal que cada nodo agregador de datos calcula el umbral T (n):

$$T(n) = \begin{cases} \dfrac{P}{1 - P\left(Rmod\left(1/P\right)\right)} \\ 0 \end{cases}$$

Aquí P es el porcentaje deseado de *cluster heads*, R es un número entero, y G es el conjunto de los nodos que no han sido *cluster heads* durante las últimas 1/P rondas. Para ser *cluster head*, un nodo n escoge un número aleatorio entre [0,1] y se convierte en una *cluster head* si este número es inferior que la T(n). En la segunda fase, los nodos envían sus datos a las *cluster heads* según la lista establecida. LEACH requiere que las *cluster heads* envíen sus datos agregados a la estación base a través de un solo eslabón. Sin embargo, esto es una desventaja porque la transmisión con un sólo eslabón puede ser bastante costosa cuando la estación base está lejos de la *cluster head*. LEACH es completamente distribuido, ya que no requiere ningún conocimiento global sobre la estructura de la red. Por otra parte, puede haber una alta sobrecarga de mensajes de control si la topología de la red es dinámica debido a los nodos móviles.

Otro protocolo de agregación de datos no jerárquico, es el llamado HEED [11]. Para la selección de la *cluster head*, presta atención a los beneficios de la disponibilidad de múltiples niveles de potencia en los nodos. Es un combinado métrico que está

compuesto por la energía residual del nodo y la proximidad del nodo a sus vecinos. HEED define el promedio del nivel de poder mínimo requerido por todos los nodos dentro del *cluster* para alcanzar la *cluster head*. Llaman a esto *Average Minimum Reachability Power* (AMRP). AMPR es usado para estimar el coste de comunicación en cada *cluster*. Para seleccionar las *cluster heads*, cada nodo calcula su probabilidad de convertirse en *cluster head* así:

$$P_{(CH)} = C \times \frac{\mathbf{Eresidual}}{\mathbf{Emax}}$$

donde C, $E_{residual}$ y E_{max} denotan el porcentaje inicial de cabezas de cluster, la corriente residual, y firman con las iniciales la energía del nodo de sensor, respectivamente. Cada nodo difunde un mensaje de *cluster head*, los nodos de sensor seleccionan su *cluster head* como el nodo con AMRP más bajo en el conjunto de mensajes recibidos. Este proceso continúa recurrentemente hasta que cada nodo sea asignado a una *cluster head*. Como en LEACH, las *cluster heads* en HEED, se comunican directamente con la estación base. Los resultados de las pruebas llevadas a cabo con HEED muestran que HEED amplía la vida de la red.

Otra propuesta es un esquema de agrupación que funciona por agregación de datos periódicamente por salto [12]. El esquema propuesto se llama Cougar y es muy útil para usos donde los nodos generan datos correlacionados continuamente. Una vez que las *cluster heads* agregan sus datos, ellos envían los datos locales agregados a un nodo de entrada. Similar a LEACH, Cougar puede verse alterado por la topología dinámica de la red. Sin embargo Cougar tiene un procedimiento de elección de la *cluster head* único. Éstas son seleccionadas basándose en varios datos medidos y permite a los nodos estar más lejos que un salto de sus *cluster head*. Para ello se necesitan algoritmos de enrutamiento para cambiar paquetes dentro de los *clusters*. Cougar emplea el Ad hoc On Demand Distance Vector (AODV). En Cougar se emplea la sincronización para agregar los datos correctamente. La *cluster head* es sincronizada con todos los nodos en el *cluster* y este no revela sus datos agregados hasta que todos los nodos envíen sus datos. Por lo tanto, el mecanismo de sincronización es el que permite la corrección de los datos agregados.

Clustered Diffusion with Dynamic Data Aggregation (CLUDDA) [13] es un enfoque híbrido que combina clustering con mecanismos de difusión. En CLUDDA cada mensaje contiene la definición de la pregunta que describe las operaciones que tienen que ser realizadas sobre los componentes para generar una respuesta apropiada. CLUDDA combina directamente la difusión y el clustering durante la fase inicial de propagación. Usando el mecanismo de clustering se garantiza que sólo el grupo de *cluster heads* que llevan a cabo la comunicación de cluster está implicado en la transmisión de mensajes de interés. Como los nodos regulares no transmiten ningún dato a no ser que ellos sean capaces de revisar una petición, CLUDDA conserva la energía. En CLUDDA, cualquier *cluster head* puede realizar la agregación de datos, y de ahí que los puntos de agregación sean dinámicos. También, cada *cluster head* mantiene una caché de consultas para conservar los diferentes datos que fueron agregados para obtener los datos finales. Las *cluster head* también guardan una lista de las direcciones de los nodos vecinos de los cuales provinieron los mensajes. Estas

direcciones se usan para propagar mensajes de interés directamente a nodos específicos en vez de usar mecanismos de difusión.

Las *cluster heads* realizan la agregación y envían los datos agregados a la estación base por caminos con múltiples saltos. Sin embargo, durante la transmisión de la información agregada desde las *cluster heads* a la estación base no se realiza ninguna agregación remota.

2.4 LA AGREGACIÓN SEGURA DE DATOS

Como cualquier otro protocolo de redes de sensores inalámbricos, los protocolos de agregación de datos deben satisfacer las exigencias de seguridad. Sin embargo, los recursos limitados de los nodos y la necesidad de datos sin formato para el proceso de agregación plantean grandes desafíos en la aplicación de la seguridad y la agregación de datos al mismo tiempo. Las exigencias de seguridad pueden ser satisfechas usando criptografía de claves simétricas o asimétricas. En muchos casos, Debido a las restricciones de los recursos, la criptografía de clave simétrica es mejor que la criptografía de clave asimétrica. La necesidad de poner en práctica la agregación de datos y la seguridad que usa algoritmos de criptografía de claves simétricas ha conducido a muchos investigadores a trabajar sobre el problema de la agregación de datos segura [14-22]. En estos protocolos, la seguridad y la agregación de datos se alcanzan simultáneamente utilizando *hop-by-hop*. Es decir, los agregadores de datos deben descifrar cada mensaje que reciben, agregar los datos según la función de agregación correspondiente y cifrar el resultado de la agregación antes del envío. Además, estos esquemas requieren agregadores de datos para establecer claves secretas con sus nodos vecinos. Por lo tanto, los protocolos de agregación de datos seguros *hop-by-hop* no pueden proporcionar la confidencialidad de los datos ni un buen resultado de latencia debido al proceso de descifrado/cifrado. Para mitigar las desventajas de *hop-by-hop*, se proponen protocolos de agregación de datos seguros [23-28]. Los protocolos propuestos realizan la agregación de datos sin requerir el descifrado de los datos en los nodos agregadores. Mientras algunos de estos protocolos usan la criptografía simétrica, otros emplean funciones de criptografía asimétrica [29-30]. Como los agregadores de datos no tienen que descifrar los datos del sensor para realizar la agregación, los protocolos propuestos proporcionan la confidencialidad de los datos y causan menos latencia comparada con los protocolos de agregación de datos seguros que usan *hop-by-hop*. Por otra parte, el inconveniente de los protocolos de agregación de datos que no requieren el descifrado de los datos es que sólo son aplicables un conjunto de funciones como la suma y el promedio.

2.4.1 La agregación segura de datos que usa datos simples

Se proponen mecanismos de seguridad para descubrir la mala conducta de algunos nodos [14]. La idea clave de este trabajo es la agregación retrasada. En vez de agregarse mensajes en el salto inmediato, los mensajes son reenviados inalterados sobre el primer salto y luego agregados en el segundo salto. Esto se consigue usando una clave difundida por la estación base. Así, los nodos tienen que proteger los datos para autentificarlos cuando la estación base difunda la clave de autentificación. El protocolo propuesto asegura la integridad de los datos, y sin embargo esto no proporciona la confidencialidad de los datos. Además, si un nodo padre y su hijo son nodos

comprometidos, entonces la integridad de los datos tampoco está garantizada. Algunos protocolos [15] como SIA utilizan mecanismos de muestreo arbitrarios y pruebas interactivas para comprobar la corrección de los datos agregados en la estación base. Los autores defienden que, utilizando estos mecanismos es posible para el usuario verificar que los datos proporcionados por el nodo agregador son una aproximación correcta incluso cuando el nodo agregador y parte de los nodos están comprometidos. Los autores presentan protocolos eficientes para calcular la media y el promedio de las medidas, estimando el tamaño de la red, y encontrando la lectura mínima y máxima. La corrección de datos es comprobada construyendo un árbol de Merkle. En esta construcción, todos los datos recogidos son colocados en las hojas del árbol, y el nodo agregador calcula un árbol binario que comienza en los nodos hoja: cada nodo interno en el árbol se calcula como el valor de los dos nodos hijos. La Figura 8 muestra un ejemplo de la construcción del árbol de Merkle. En este protocolo se asume que cada nodo tiene un identificador único y comparte una clave secreta criptográfica con la estación base y con el agregador. Estas claves permiten la confidencialidad, la integridad y la autenticación de los datos.

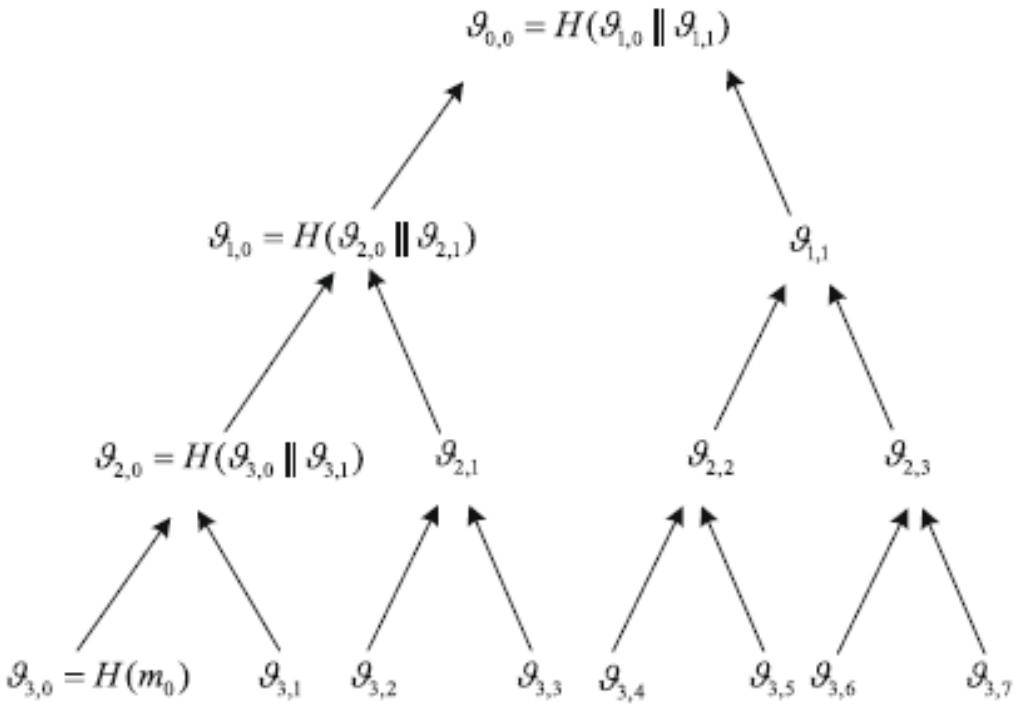


FIGURA 8: Un ejemplo de construcción de un árbol de Merkle.

El protocolo SecureDAV [16] es muy similar a los llamados SIA [15] pero usa criptografía de curva elíptica para el cifrado. Además, SecureDAV mejora la integridad de los datos firmando los datos agregados. Todos los nodos forman parte de un *cluster* y comparten una clave de *cluster* secreta. Cada nodo es capaz de generar una firma parcial sobre los datos agregados. Cada nodo agregador agrega sus datos de *cluster* y difunde los datos agregados a su *cluster*. Cada nodo en el *cluster* compara sus datos con los datos agregados difundidos por el nodo agregador. Un nodo firma parcialmente los datos agregados si la diferencia entre sus datos y los datos agregados es menos que un

umbral. Finalmente, los nodos agregadores combinan las firmas parciales para formar una firma completa de los datos agregados y lo envían a la estación base. SecureDAV proporciona la confidencialidad de los datos, la integridad de los datos, y la autentificación de la fuente. Sin embargo, el sistema supone una sobrecarga de comunicación sobre la validación de los datos y sólo es compatible con la función de agregación promedio.

Algunos algoritmos como SAT [19], sólo usan la criptografía cuando descubren que alguno de los nodos no funciona correctamente. En dicho algoritmo, cualquier nodo hijo es capaz de escuchar los datos que llegan a su nodo padre. Cuando los datos agregados son cuestionables se emplea un esquema de votación ponderado para decidir si los datos agregados son con confiables o no. Si los nodos agregadores no son confiables, entonces SAT se reconstruye de modo que se excluyan del árbol de agregación.

SDAP [21] es un protocolo de agregación de datos seguro que utiliza la técnica *hop-by-hop*. Los autores de este algoritmo muestran interés en el hecho de que algunos nodos de alto nivel posean mayor grado de confianza durante el proceso de agregación que otros nodos de menor nivel. Los datos agregados calculados por un nodo de alto nivel representan a los datos de un número grande de nodos de bajo nivel. Si un nodo comprometido está más cerca de la estación base, los datos falsos agregados por este nodo comprometido tendrán un mayor impacto sobre el resultado final calculado por la estación base. Dado que todos los nodos tienen un hardware simple que es propenso a poder ser comprometido, ninguno de los nodos de bajo nivel debería ser más confiable que otros. SDAP tiene el objetivo de reducir la confianza sobre los nodos de alto nivel siguiendo el principio divide y vencerás. SDAP divide dinámicamente el árbol en múltiples grupos lógicos (los subárboles) de tamaños similares que usan una aproximación. De este modo, menos nodos son localizados bajo un nodo de alto nivel en un subárbol lógico que causa la amenaza a la seguridad por un nodo de alto nivel comprometido. En la Figura 9 se muestra un ejemplo de un árbol agrupado. SDAP proporciona la confidencialidad de los datos, la autenticación de la fuente, y la integridad de los datos.

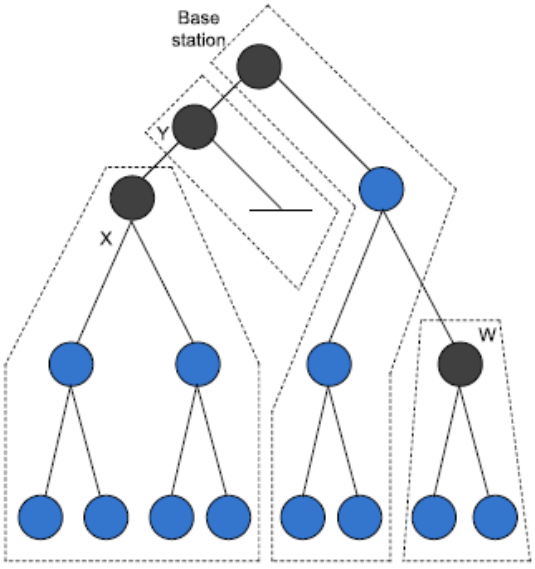


FIGURA 9: Un ejemplo del árbol de agregación en SDAP.

En [22] se argumenta que los nodos comprometidos tienen acceso a las claves criptográficas que se emplean para asegurar el proceso de agregación y por lo tanto por sí solas no pueden proporcionar una solución suficiente para el problema de la agregación de datos. Basándose en esta observación, los autores proponen un protocolo de agregación de datos seguro y confiable llamado SELDA. La idea básica que está detrás de SELDA es que los nodos observen las acciones de sus nodos vecinos para desarrollar niveles de confianza. Como se muestra en la Figura 10, los nodos emplean mecanismos de vigilancia para descubrir la disponibilidad de cada nodo y las malas conductas de sus vecinos. Estas malas conductas son cuantificadas en niveles de confianza que usan la función de distribución Beta [31-32]. Los nodos cambian sus niveles de confianza con los nodos vecinos para formar una red de confianza que permite a los agregadores diseñar caminos seguros. Los resultados de la simulación muestran que SELDA aumenta la fiabilidad de los agregadores. Después se mejoró la idea principal de SELDA introduciendo el concepto de reputación funcional donde cada valor de reputación funcional es calculado sobre las acciones de cada nodo en lo que concierne a dicha función. Por tanto, la seguridad en el proceso de agregación de datos está garantizada por la selección de datos de confianza en los nodos agregadores gracias a la utilización de la reputación funcional y por la ponderación utilizando los datos del sensor de detección de reputación funcional. Los resultados de la simulación muestran que la utilización de la reputación funcional es más eficaz que la utilización de la reputación general evaluando la honradez de un nodo.

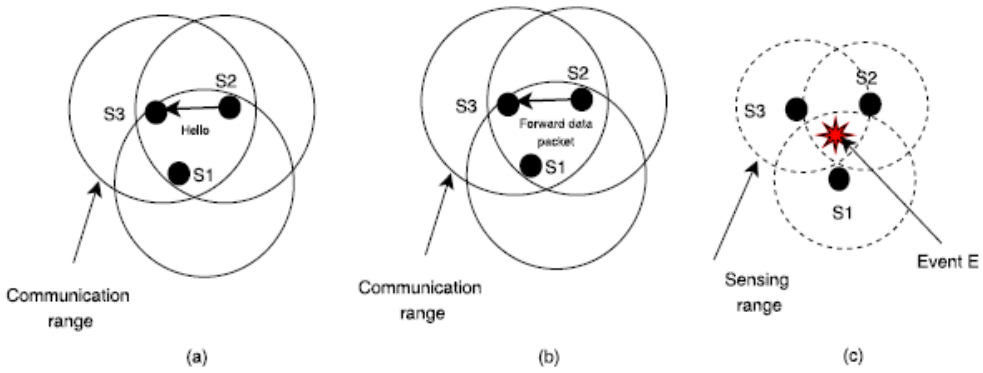


FIGURA 10: La figura muestra un ejemplo de cómo detectar el mal comportamiento de un nodo.

En redes de sensores inalámbricos, un nodo comprometido puede inyectar datos falsos durante la agregación dañando la integridad de los datos agregados. Es muy útil para los nodos descubrir los datos falsos cuanto antes para evitar agotar sus recursos como por ejemplo la batería. Aunque varios protocolos de agregación de datos seguros sean capaces de descubrir los datos falsos inyectados por los nodos, las inyecciones de datos falsos por lo general son difíciles de descubrir. La razón es que la agregación de datos causa alteraciones en los datos y por lo tanto un cambio en los datos agregados debido a la inyección de datos falsa es sumamente difícil de descubrir. Dichas inyecciones de datos falsos por agregadores de datos comprometidos fácilmente pueden causar falsas alarmas y el gasto de los recursos de la red reduciendo la eficacia operacional.

En algunos casos, el problema de la agregación segura de datos se estudia desde la perspectiva de la detección de intrusos [33-34]. Desde este punto de vista se propone el Extended Kalman Filter (EKF), un mecanismo para descubrir datos falsos inyectados. Con el empleo de EKF, se supervisan los nodos para predecir sus verdaderos valores futuros en la red. Usando funciones de agregación diferentes (el promedio, la suma, el máximo y el mínimo), los autores muestran cómo obtener gamas normales. Por otra parte, también se muestra que EKF se utiliza para crear mecanismos eficaces de detección que son capaces de distinguir entre acontecimientos malintencionados y eventos de emergencia y por lo tanto esto puede reducir la tasa de falsa alarma en la red. Los resultados de la simulación demuestran que las técnicas propuestas alcanzan el funcionamiento deseable para descubrir datos falsos inyectados.

Otros estudios [35] analizan el hecho de que muchas técnicas de detección de datos falsos existentes consideran inyecciones de datos falsas sólo durante el proceso de transmisión. Se presenta un protocolo de agregación de datos llamado DAA, para integrar la detección de datos falsos con la agregación de datos y la confidencialidad. Para apoyar la agregación de datos con la detección de datos falsos, proponen un algoritmo de supervisión. Para asegurar que la transmisión de información sea confidencial, los nodos entre dos agregadores de datos consecutivos verifican la integridad de los datos sobre los datos cifrados. Cada paquete de datos se añade con dos códigos de autentificación de mensaje. El análisis del funcionamiento muestra que DAA descubre cualquier dato falso inyectado y que los datos falsos descubiertos no son enviados más allá de los siguientes agregadores de datos. A pesar de esta detección de datos falsos y el aumento de la confidencialidad de los datos, los resultados de la simulación muestran que DAA todavía puede reducir la cantidad de datos transmitidos hasta el 60 % con la ayuda de la agregación de datos y la detección y eliminación temprana de datos falsos en el camino.

Hay estudios acerca de cómo determinar un umbral de alarma falso de forma dinámica y eficaz a fin de reducir al mínimo la probabilidad de falsas alarmas en una red de sensores inalámbricos [36]. El umbral cambia de forma dinámica por lo que se consigue mejor probabilidad de detección y la reducción del número de alarmas falsas. Se propone reducir el impacto tomando un promedio ponderado de lecturas diferentes para la misma medida. Aunque la confidencialidad de los datos y la autentificación no sean consideradas en el algoritmo de agregación de datos propuesto, los resultados que se obtuvieron con la prueba de este algoritmo muestran que esto mejora la exactitud de la detección de intrusos y reduce al mínimo la tarifa de alarmas falsas en la red.

La mayoría de los protocolos relacionados con la seguridad de los datos usan datos reales para la agregación y por ello, se necesita el desciframiento de los datos en los nodos agregadores. Sin embargo, también se proponen protocolos [17-20] que no necesitan datos reales y por lo tanto son capaces de integrar la seguridad y la agregación de datos sin problemas. Se presenta el modelo de la agregación de datos basado en la energía eficiente (ESPDA) [17] que considera al mismo tiempo la agregación de los datos y su seguridad. ESPDA es el primer protocolo para considerar técnicas de agregación de datos sin comprometer la seguridad. ESPDA usa códigos para realizar la agregación de datos. Los códigos se extraen de los datos reales de tal modo que cada código tiene las características de los datos reales correspondientes. El proceso de extracción puede variar dependiendo del tipo de dato. Por ejemplo, cuando los datos reales son las imágenes de seres humanos captados por los sensores de vigilancia, los

valores de los parámetros claves para la cara y el reconocimiento del cuerpo se consideran los datos representativos dependiendo de las exigencias de la aplicación. Las *cluster heads* determinan los distintos códigos y luego solicitan que un nodo envíe los datos reales para cada código distinto. Este enfoque hace que ESPDA sea eficaz tanto desde el punto de vista de la energía como de la amplitud de banda. ESPDA es también seguro porque las *cluster heads* no tienen que descifrar los datos para la agregación de datos y no se difunde ninguna clave de cifrado/descifrado.

Tras esto, aparece SRDA (Secure Reference-Based Data Aggregation) [20]. Como ESPDA, SRDA también asume el hecho de que los protocolos de agregación de datos deben trabajar con los protocolos de seguridad de comunicación de datos, y que cualquier conflicto entre estos protocolos podría perjudicar a la seguridad de la red. En SRDA, los datos brutos medidos por los nodos son comparados con valores de referencia y sólo se transmiten las diferencias en los datos. Los datos de referencia se toman como el valor medio de un número de lecturas anteriores. Es muy importante reducir el número de bits en una transmisión porque la comunicación de radio es la actividad que consume la mayor parte de la energía en un nodo. Mientras la agregación de datos reduce el número de paquetes, disminuyendo el tamaño de los paquetes transmitidos se mejorará el ahorro de energía. En algoritmos de agregación de datos convencionales, los sensores transmiten sus datos a las *cluster heads*. Sin embargo, SRDA transmite los datos diferenciales en vez de todos los datos leídos. Por consiguiente, la agregación diferencial reduce la cantidad de datos transmitidos desde los nodos a las *cluster heads*. El inconveniente de ESPDA y SRDA es que no permiten realizar la agregación de datos a nodos intermedios. Es decir, los datos pueden ser agregados sólo en agregadores de datos, lo que limita considerablemente la ventaja de la agregación de datos.

2.4.2 Agregación de datos segura que usa datos cifrados

Usando algoritmos de criptografía de claves simétricas no es posible alcanzar la confidencialidad de extremo a extremo y la agregación de datos al mismo tiempo. Si el uso de algoritmos de criptografía de claves simétricas es combinado con las exigencias de la agregación de datos eficiente, entonces los mensajes deben ser cifrados *hop-by-hop*. Sin embargo, esto significa que, para realizar la agregación de datos, los nodos intermedios tienen que descifrar cada mensaje recibido, luego agregar los datos según la función de agregación correspondiente, y finalmente cifrar el resultado de la agregación antes de su transmisión. Esto no es una forma eficiente de administrar la energía y puede causar un retraso considerable. Además, este proceso requiere de los agregadores de datos vecinos para compartir claves secretas para el descifrado y el cifrado. Para alcanzar la confidencialidad de los datos de extremo a extremo y la agregación de datos al mismo tiempo sin requerir la clave secreta que comparten los agregadores de datos se han llevado a cabo numerosos estudios. [23-27]

Un homomorfismo de aislamiento es una transformación de cifrado que permite el cómputo directo sobre los datos cifrados. E denota el cifrado y D denota el descifrado. Al mismo tiempo, + denota la adición y * denota la operación de multiplicación sobre un conjunto Q de datos. Kpr y Kpu son las claves privada y pública de la estación base. Una transformación de cifrado se acepta como homomórfica aditiva si

$$a + b = DKpr\ (EKpu\ (a) +\ (EKpu(b))$$

Y se acepta multiplicativa homomórfica si

$$a * b = DKpr\ (EKpu\ (a) *\ (EKpu(b))$$

En la agregación de datos oculta (CDA) [23], los nodos comparten una clave común con la estación base que se mantiene oculta a los agregadores de datos intermedios. La contribución principal de CDA es la provisión de cifrado de extremo a extremo para el tráfico de multidifusión inverso entre los sensores y la estación base. En el enfoque propuesto, los agregadores de datos realizan las funciones de agregación que son aplicadas a los datos cifrados. Esto proporciona la ventaja de que los agregadores de datos intermedios no tienen que realizar el costoso desciframiento y las operaciones de cifrado. Por lo tanto, los agregadores de datos no tienen que almacenar las claves criptográficas.

Un protocolo de agregación de datos seguro llamado CDAP [27] aprovecha la privacidad de la clave asimétrica basada en la criptografía homomórfica para alcanzar la confidencialidad de los datos de extremo a extremo y la agregación de los datos al mismo tiempo. La criptografía asimétrica genera una sobrecarga computacional que no está al alcance de todos los nodos debido a sus recursos limitados. Para mitigar este problema, el protocolo CDAP emplea un conjunto de nodos con mayor cantidad de recursos llamados nodos agregadores (AGGNODEs), para el cifrado homomórfico y la agregación de los datos cifrados. En CDAP, después del despliegue de la red cada AGGNODE establece claves con sus nodos vecinos de modo que los nodos vecinos puedan enviar sus lecturas. En la fase de colección de datos del protocolo CDAP, cada AGGNODE pregunta a sus nodos vecinos. Cada nodo vecino cifra sus datos (usando el algoritmo RC5) y envía los datos cifrados a su AGGNODE. El AGGNODE descifra todos los datos recibidos de sus vecinos, los agrega, y cifra los datos agregados. Una vez que los datos son cifrados con el algoritmo de cifrado homomórfico, sólo la estación base puede descifrarlos usando su clave privada. El AGGNODE intermedio puede agregar aquellos datos cifrados durante la expedición de datos. Por lo tanto, los datos recogidos por los nodos son agregados por AGGNODES en su ruta a la estación base. La estación base descifra los datos agregados que usan su clave privada. La Figura 11 es un ejemplo ilustrativo de agregación de datos en CDAP. Durante la fase de colección de datos inicial del protocolo CDAP, los nodos usan algoritmos de cifrado con claves simétricas. Debido al cifrado simétrico, un AGGNODE comprometido puede revelar los datos de sus nodos vecinos o inyectar datos falsos en los datos. Sin embargo, los autores argumentan que el efecto de este ataque es local.

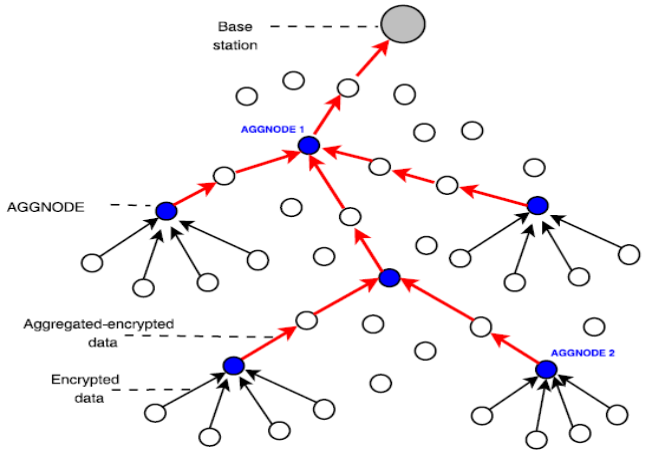


FIGURA 11: Un escenario de agregación de datos siguiendo el protocolo CDAP.

Más tarde, las investigaciones dejan ver que los protocolos homomórficos existentes sólo pueden trabajar para algunas funciones de agregación específicas. Por ello, los autores proponen un enfoque estadístico para la autentificación de datos. La nueva idea es modular la información de autentificación y superponer esta información sobre los datos sensoriales en los nodos. La técnica propuesta, visualiza los datos sensoriales de la red entera en un instante de tiempo como una imagen, en la cual cada nodo se ve como un pixel con su lectura sensorial que representa la intensidad de pixeles. Para equilibrar el consumo de energía entre los nodos, se utiliza la técnica DSSS. El esquema propuesto adopta tanto los esquemas de compresión de imagen existentes como las funciones de agregación para reducir la carga de la red conservando los datos. Además, usando un esquema DSSS, permiten a la técnica propuesta sobrevivir a un cierto grado de distorsión y por lo tanto apoyar la agregación de datos.

Protocolo	Confidencialidad de los datos	Integridad de los datos	Autentificación de la fuente	Disponibilidad de los nodos
Hu et al. [14]		✓	✓	
SIA [15]	✓	✓	✓	
SecureDAV [16]	✓	✓	✓	
ESPDA [17]	✓	✓	✓	
Du et al. [18]		✓	✓	
Wu et al. [19]		✓	✓	
SRDA [20]	✓	✓	✓	
SDAP [21]	✓	✓	✓	
SELDA [22]		✓	✓	✓
Ozdemir [37]		✓	✓	✓
CDA [23]	✓			
Castelucia et al. [24]	✓			
Ozdemir [27]	✓			
Zhang et al. [28]		✓		
Rodhea et al. [26]	✓			
Sun et al. [33,34]		✓	✓	
DAA [35]	✓	✓	✓	

TABLA1: Comparación de los diferentes protocolos de agregación de datos seguros.

En la Tabla 1 se recoge la comparación de los diferentes esquemas de agregación de datos seguros. Casi todos los protocolos de agregación de datos seguros garantizan la integridad de datos y la autentificación de la fuente. Los protocolos [23,24,26,27] únicamente se centran en la agregación de datos cifrados y no proporcionan la integridad de los datos y la autentificación de la fuente. Sin embargo, estos protocolos pueden ser modificados fácilmente para incluir la integridad de los datos y la autentificación de la fuente. La Tabla 1 muestra también que algunos protocolos de agregación de datos seguros [14,18,19,22,28,37] no tienen en cuenta la confidencialidad de los datos que es esencial. Por lo tanto, estos protocolos deberían utilizarse sólo en usos en los cuales los datos transmitidos no sean secretos. Entre los protocolos que proveen la confidencialidad de los datos, se encuentran los protocolos [17,20,23,24,27,35]. Sólo los protocolos de agregación de datos [22,37] incluyen el estudio de la disponibilidad de los agregadores de datos en donde se utilizan mecanismos de supervisión extensos. Considerando que la confidencialidad de los datos, la integridad de éstos y la autenticación de la fuente son las exigencias de seguridad más importantes, podemos concluir de la Tabla 1 que los protocolos de agregación de datos propuestos en [15-17,20,21,35] proporcionan la mejor seguridad comparada con otros protocolos de agregación de datos.

2.5 LA INVESTIGACIÓN ABIERTA Y FUTURAS INVESTIGACIONES

Aunque se han planteado muchas soluciones para los problemas de agregación de datos, hay todavía mucha investigación por llevar a cabo, sobre todo desde el punto de vista de la seguridad. La relación entre los mecanismos de enrutamiento y los protocolos de agregación de datos ha sido muy estudiada ya que son temas sumamente relacionados. Se han propuesto muchos protocolos de agregación de datos no jerárquicos. Aunque, estos protocolos han demostrado ser muy eficientes en redes estáticas en las cuales los *clusters* no cambian para un tiempo suficientemente grande, en redes dinámicas se llevan a cabo bastante mal. Una posible línea para futuras investigaciones es la agregación de datos en ambientes dinámicos. El impacto de la heterogeneidad de los nodos sobre los protocolos de agregación de datos es otra área de investigación inexplorada. Los protocolos que usan nodos de mayores prestaciones como agregadores de datos presentan resultados prometedores.

La seguridad es una cuestión importante para el proceso de agregación de datos y tiene que seguir siendo investigada. Uno de los principales problemas es que los agregadores de datos comprometidos pueden inyectar datos falsos. Como la agregación de datos por lo general causa alteraciones en los datos, las inyecciones de datos falsas que se llevan a cabo por nodos agregadores de datos comprometidos son difíciles de descubrir. Las técnicas propuestas se basan en mecanismos de control muy amplios. La eficacia de estos protocolos de supervisión no está totalmente evaluada y por lo general provocan el consumo de los recursos del sensor. Por todo esto, el desarrollo de mecanismos de supervisión para el proceso de agregación de datos seguro es un camino interesante para futuras investigaciones.

Para proporcionar la seguridad extremo a extremo, los protocolos de agregación de datos seguros basados en homomorfismos han llamado recientemente la atención

considerablemente. Sin embargo, el diseño y la puesta en práctica de este tipo de agregación deben ser todavía investigados. Existen muchos protocolos que utilizan la criptografía de la clave pública pero no es posible que lleguen a buen puerto debido a la limitación de recursos de los sensores. Es por esto, por lo que en algunos protocolos de agregación de datos se utiliza la criptografía de curva elíptica las cuales sólo pueden trabajar para algunas funciones de agregación específicas a base de preguntas, por, ejemplo la suma, el promedio... Por lo tanto, el diseño eficiente de las funciones homomórficas que son capaces de trabajar con todos los tipos de funciones de agregación de datos debe ser explorado.

Además, a la aplicación de la teoría de la codificación de la fuente de agregación de datos se le ha dedicado muy poca atención hasta ahora. Considerando que los datos están sumamente relacionados, la agregación de datos puede alcanzarse utilizando estas técnicas. La investigación existente en esta área sólo da como resultado casos teóricos y ninguno de estos es aplicable todavía a redes de sensores inalámbricos reales. Además, no hay ningún protocolo de agregación de datos seguro que use la idea de codificación de la fuente que integre la confidencialidad de datos y la agregación al mismo tiempo.

Con respecto a la agregación de datos jerárquica se podrían producir cuantiosas investigaciones en el futuro.

3. DISEÑO DEL PROYECTO

En este capítulo se explica el proceso de creación del proyecto lo que nos permitirá comprobar la utilidad del algoritmo que se ha desarrollado.

3.1 ANÁLISIS DEL PROYECTO

El objetivo principal en el desarrollo de este proyecto es conseguir un algoritmo para la comunicación segura en una red de sensores inalámbricos. En la primera parte del algoritmo los nodos intercambiarán varios mensajes con el fin de llevar a cabo un proceso que podemos llamar de registro en el que formarán la estructura de la red descubriendo quiénes serán sus vecinos y mandando esa información al nodo que actuará como estación base para que éste también tenga toda la información registrada. La segunda parte consistirá en otro intercambio de mensajes pero esta vez con la información cifrada para asegurar que sólo puede conseguir la información el nodo hacia el que va dirigida.

Al comienzo del desarrollo el uso de criptografía de curva elíptica (ECC) [39] parecía la más adecuada, ya que al tratarse de nodos de redes inalámbricas de sensores con recursos limitados, era necesario utilizar las mínimas operaciones de cómputo y claves que ocupasen menos espacio (160bits). Tras comenzar con la implementación se descubrió que el *Micro Framework* del sensor imote2 no soportaba dicho algoritmo de criptografía por lo que nos decantamos por RSA [40], un algoritmo que toma su nombre de sus creadores Rivest [41], Shamir [42] y Adleman [43] y que nos aporta seguridad similar a ECC pero con un mayor consumo de recursos ya que sus claves ocupan 1024 bits.

El principal uso para el que se pensó este algoritmo es para el transporte inteligente de mercancías. Al ser un caso de transporte intermodal, esta red podrá estar transportada por diferentes medios, tales como camiones, trenes, barcos o aviones. El sensor o nodo, será la unidad mínima de carga, en este caso un contenedor estándar. Hay que tener en cuenta que los contenedores de diferentes compañías coexistirán en un mismo medio de transporte o lugar de almacenaje. Es por este hecho por el que la criptografía toma tanta importancia.

3.2 PROBLEMAS EN LA IMPLEMENTACIÓN DEL ALGORITMO

Al comienzo de la implementación la mejor idea parecía utilizar hilos para hacer que los sensores pudiesen estar esperando mensajes y enviándolos al mismo tiempo pero tras varios intentos de llevar este objetivo a cabo y debido a los problemas que se encontraron, la idea fue desechada. El principal motivo por el que no se pudo desarrollar este planteamiento fue que debía haber una instancia de la clase radio escuchando mensajes y otra enviando ya que si se utiliza una para las dos actividades los mensajes colisionan y tampoco se consigue el objetivo pero duplicar la radio en un mismo programa daba muchos problemas. Es por este motivo por el cual el algoritmo no utiliza hilos sino que realiza las tareas que tiene que llevar a cabo de forma secuencial.

Otro de los problemas encontrados al principio del desarrollo fue que tan sólo podíamos contar con dos sensores lo que por una parte simplificaba el trabajo pero al mismo tiempo cambiaba las ideas que en un principio se habían desarrollado para programar el algoritmo obligando a hacer algunos cambios en el mismo.

El método *ToXmlString()* [44] de la clase *RSACryptoServiceProvider* [44] es el más utilizado para generar aleatoriamente claves RSA públicas y privadas que luego podrían ser utilizadas por el algoritmo para la codificación de los datos. Se creó un programa generador de claves que utilizaba esta clase y la clase *StreamWriter* [46] para escribirlas en un fichero. Si al método *ToXmlString()* [44] se le pasa el parámetro falso, se obtiene una clave pública generada de forma aleatoria mientras que si se utiliza el parámetro verdadero, devuelve de igual manera una clave pública seguida de su correspondiente clave privada. Las siguientes líneas representan una de las claves que se guardaron en el fichero tras utilizar el programa para generar las claves.

Si utilizamos el método *ToXmlString()* [44] con el parámetro falso:

<RSAKeyValue><Modulus>2t9atP0DLWblRRF6b21rjMZIWLS6LzwKcwBPC76ltZ+/r9VPV
uUNr1FF5UocNlI66Lmchc7WOfY6lARPkRiY16yAlHpR+4rKUarx7vZJ06kSgdYLA9H1z28KPubTh10
Re3420oPmi28nha7eUSaZl5pvw5tqHbh3wQjOxpxXgV0=</Modulus><Exponent>AQAB</Expon
ent></RSAKeyValue>

Si utilizamos el método *ToXmlString()* [44] con el parámetro verdadero:

<RSAKeyValue><Modulus>2t9atP0DLWblRRF6b21rjMZIWLS6LzwKcwBPC76ltZ+/r9VPV
uUNr1FF5UocNlI66Lmchc7WOfY6lARPkRiY16yAlHpR+4rKUarx7vZJ06kSgdYLA9H1z28KPubTh10
Re3420oPmi28nha7eUSaZl5pvw5tqHbh3wQjOxpxXgV0=</Modulus><Exponent>AQAB</Expon
ent><P>8+3dUolxM1MQ3vEfhA+t6lw1i6LznmGTMzWIfRvs1fnBt1i489rE94W+gk2lfdePQTDTPE
QMg/KbvMxMKhhvzw==</P><Q>5bQRS0sxm43YUgzeW9GypMjqAhMy8HxTOKhQ04iArvE9JBg
1eL1v44ipQOsVvNB0fDbSFmW2m1P7m7ih8zW7Ew==</Q><DP>dqhW6TeCoXze0BRrSEp4R3Dj
ezGhYIIGoPqBkppq8wuzeATos3bQKbgdnDU5M6YQQZ4Go2JW1DjIGWcoBbItcQ==</DP><DQ>Z
v8IyiQbSE6f8pqGkdFDMxRLqUaGjsKQZlVCeLc6TDcrXSXlxA/dPb8ndU5z3sfxk0jZhuVwrdFIfpiZN
X6gNQ==</DQ><InverseQ>HMndz29BxNa3eZxIeC0Ql/sXFMK7UCW1L+EE0ITGCf4gIV+tfUFQN8
2jgrMhSLnNAuczmnNwBs4f+iGWpKFLzg==</InverseQ><D>VfJVRTMBd2erajbV1EGsdQg7ypGV
dwzlsYkSgTMpESwc8qgZ0q55dDm+763tbCtYNKEGZMj2+KXHFO6oa/qzsWtvwMPXeOu5WSxib
W52L5Sj0e9DW+uHzRlQ6sdR0Z0+Z7znZoz8VyUqw8QQLtIK+BxYnsPEvQ7g7lTqephQC2U=</D>
</RSAKeyValue>

Se puede comprobar cómo el primer *string* devuelto por el método coincide con la primera parte del segundo ya que se trata de la clave pública. Hasta aquí no se planteó ningún problema porque este programa se ejecutaba de manera independiente a los sensores y por tanto al *Micro Framework* de estos. Era imprescindible utilizar otros métodos de la clase *RSACryptoServiceProvider* [44] para cifrar y descifrar información si se habían obtenido las claves mediante dicha clase. Tras varias pruebas quedó patente la incompatibilidad de la clase con el *Micro Framework* que utilizan los sensores imote2 [3] por lo que hubo que desechar el programa creado para generar las claves y buscar otra manera de utilizar criptografía RSA.

Otro de los detalles que han retrasado el trabajo es que los paquetes que pueden mandar los sensores tienen un tamaño limitado, lo cual no se tuvo en cuenta al principio

por lo que los mensajes no se enviaban correctamente o la información que transmitían no era la adecuada sin motivos aparentes. Este detalle ralentizó mucho el proceso ya que no era fácil de apreciar por qué sucedía esto. Tras encontrar el problema fue necesario enviar la información que ocupaba más de 90 bytes en varios paquetes.

El último problema encontrado hacía imposible descifrar información en un nodo utilizando RSA ya que la única clase que lleva a cabo esta tarea y era compatible con el Micro Framework no funciona bien en este tipo de dispositivos. El código que intenta utilizar esta manera de descifrar se adjunta y se explica a continuación así como la alternativa buscada.

3.3 GENERAR LAS CLAVES

Antes de comenzar con la ejecución del programa es necesario generar una serie de claves públicas y privadas que serán utilizadas para cifrar y descifrar los datos.

Como ya se ha comentado, el método *ToXmlString()* [44] de la clase *RSACryptoServiceProvider* [44] es el más utilizado para generar aleatoriamente claves RSA públicas y privadas que luego podrían ser utilizadas por el algoritmo para la codificación de los datos pero dicha clase no es compatible con el *Micro Framework* que utilizan los sensores imote2 [3]. Es por esto por lo que no se pueden crear las claves con una aplicación del *Micro Framework* sino que debemos utilizar la herramienta de consola *MetaDataProcessor.exe* y después incluir las claves creadas en el resto de programas. Se puede encontrar esta aplicación en el directorio de herramientas (Tools) de *.NET Micro Framework.*

Los comandos que se deben introducir en el símbolo del sistema para crear un par de claves pública y privada son los siguientes:

- cd "C:\ Archivos de programa\Microsoft .NET Micro Framework\v2.0.3036\Tools"
- metadataprocessor –create_key_pair c:\private.bin c:\public.bin
- metadataprocessor –dump_key c:\private.bin >> c:\private.txt
- metadataprocessor –dump_key c:\public.bin >> c:\public.txt

Este proceso se puede repetir tantas veces como sea necesario dependiendo del número de claves que se necesiten. Tan sólo es necesario cambiar el nombre de los directorios y los ficheros para que las claves no se sobrescriban unas sobre otras. En el caso concreto que se está tratando el proceso se repetirá dos veces obteniendo los ficheros adjuntados en la carpeta claves. De esta manera, los archivos private1.txt y private2.txt contienen el módulo y el exponente privado de dos claves diferentes mientras que los archivos public1.txt y public2.txt almacenan el módulo y el exponente público de dos claves públicas. El módulo y el exponente de estas claves siempre tienen 128 bytes de longitud. El módulo en las claves pública y privada en un mismo par es idéntico y el exponente público es siempre la misma constante. Las claves se generan al azar por lo que son únicas y si se repite el proceso varias veces se obtienen claves diferentes.

Las siguientes líneas representan un ejemplo de una de las claves privadas obtenidas tras el proceso:

```
//typedef struct tagRSAKey
//{
//   DWORD exponent_len;
//   RSABuffer module;
//   RSABuffer exponent;
//} RSAKey, *PRSAKey;
RSAKey myKey =
{
 0x00000020,
{

  0xd7,  0xc7,  0xca,  0xd8,  0x2d,  0xff,  0x0a,  0x3e,  0x8c,  0x1c,  0x2c,  0xb2,  0x99,
0x9e,  0xdb,  0xf6,  0xb9,  0x25,  0xe0,  0x77,  0xd5,  0xa6,  0x51,  0x67,  0x98,  0xac,
0x9d,  0xd8,  0xec,  0x9a,  0x9a,  0xdc,  0x1e,  0xfd,  0x75,  0xb6,  0x7c,  0x1e,  0x31,
0x15,  0xf0,  0xf2,  0x59,  0x85,  0xef,  0xdc,  0x50,  0xc4,  0x31,  0xf3,  0x88,  0xb5,
0xa4,  0xe9,  0x1e,  0xf0,  0x1b,  0xdc,  0x14,  0x70,  0xf2,  0x23,  0x55,  0x21,  0x3a,
0xe5,  0xaf,  0xa0,  0x75,  0x22,  0xe2,  0xce,  0x0f,  0x55,  0x88,  0xe0,  0x02,  0x41,
0x30,  0x6f,  0xc2,  0xd8,  0xd5,  0xa0,  0x27,  0x15,  0x63,  0xd2,  0xfa,  0x00,  0x07,
0x67,  0x23,  0x3a,  0x98,  0x9d,  0xf1,  0x5f,  0xa8,  0x03,  0xf9,  0xbb,  0x51,  0x79,
0xc4,  0x90,  0x38,  0xb9,  0x5f,  0x1e,  0xef,  0x7d,  0x08,  0xd5,  0xa9,  0x83,  0xe1,
0x9d,  0x46,  0x02,  0x21,  0x38,  0xb2,  0x84,  0xe0,  0xcf,  0x8a,  0xaf,
},
{
  0xe1,  0xcc,  0xcb,  0x41,  0x66,  0xf1,  0x43,  0x49,  0xd3,  0x0f,  0x62,  0xc5,  0x02,
0xc7,  0xc7,  0xcd,  0x9e,  0x24,  0xcd,  0xe6,  0x26,  0x55,  0xb3,  0xc4,  0xe4,  0xc3,
0xe6,  0x18,  0x46,  0x68,  0x79,  0xd3,  0x3b,  0xb9,  0x1c,  0xf2,  0x7f,  0xbd,  0xc7,
0x96,  0x99,  0xe6,  0x4c,  0xdc,  0xb9,  0xa3,  0xdb,  0x4b,  0xd9,  0x2f,  0xb7,  0x43,
0x5d,  0x39,  0xd4,  0xa3,  0xd2,  0x28,  0x04,  0x3b,  0xf8,  0x8e,  0xf7,  0x67,  0x69,
0x24,  0xff,  0x99,  0x21,  0xee,  0x56,  0xac,  0x48,  0x1b,  0xed,  0x21,  0x78,  0x2b,
0x1c,  0xee,  0xd6,  0xec,  0x80,  0x4d,  0xe8,  0xc7,  0xa8,  0x43,  0x6e,  0xef,  0x70,
0x83,  0x2b,  0x58,  0xf2,  0x4c,  0x14,  0x04,  0xc5,  0xed,  0x6a,  0xeb,  0x7e,  0x43,
0x1d,  0x7e,  0x9b,  0x5e,  0xfd,  0xd1,  0xe1,  0x6e,  0xd9,  0x40,  0xc2,  0xb5,  0xaf,
0xde,  0xeb,  0x8f,  0xdf,  0xbb,  0x1f,  0x9a,  0x11,  0x3f,  0x3d,  0x02,
},
};
```

Aunque para las claves públicas se generan archivos iguales al anterior tan sólo son necesarias tres instrucciones para conseguir la clave pública asociada a una privada ya que simplemente es necesario inicializar el exponente público porque el módulo es común a ambas. El exponente público siempre es un *string* de 128 bytes de los cuales son todos ceros excepto el primero y el tercer byte que son 0x01. Para ello se utiliza:

- static byte[] expopublico = new byte[128];
- expopublico[0] = 0x01;
- expopublico[2] = 0x01;

3.4 BASE

La estación base además de hacer de *router* entre el sistema gestor, recopilador de datos, y la red inalámbrica, en nuestro caso hará de entidad certificadora. Por esto se ha creado la clase base en la cual se llevan a cabo diferentes tareas.

En primer lugar cabe destacar que la estación base conoce las claves pública y privada de todos los nodos asociados a la red y las guarda en una lista en la que se relacionan los identificadores de cada nodo junto con su clave pública y su clave privada. Tras un periodo de registro en el que el nodo intercambia varios mensajes con la estación base, ésta envía al nodo las claves públicas de su vecino más cercano, lo que implica enviar tan sólo el módulo ya que aunque para encriptar también es necesario el exponente público, es común a todos los nodos con lo que se evita aumentar el tamaño del paquete innecesariamente. Para este proceso se necesitan mandar dos paquetes ya que el módulo de la clave tiene un tamaño de 128 bytes y los paquetes no admiten dicha longitud.

El segundo paso más importante que lleva a cabo la base es desencriptar los paquetes cifrados que recibe. Para ello utiliza la clave pública y la clave privada del nodo que recibe el paquete mediante el método *Decrypt()* de la clase *Key_RSA* [45]. Por último se muestran los resultados obtenidos como prueba de que todo el proceso funciona correctamente. Al comienzo de la implementación la mejor idea era guardar estos datos en un fichero mediante la clase *StreamWriter* [46] pero tras varias pruebas quedó patente la incompatibilidad de esta clase con el M*icro Framework* que utilizan los sensores imote2 [3], por lo que se utiliza la función *print* de la clase *SerialDump* asociada a dichas motas.

3.5 NODO

El sensor que no actúa como estación base utiliza la clase nodo para llevar a cabo su tarea. En primer lugar esta clase intercambiará una serie de mensajes con la estación base con el objetivo de recibir la clave pública de su vecino. Para ello, deberá almacenar el identificador y la dirección de su vecino más próximo que es al que enviará los mensajes; en el escenario concreto en el que se desarrolla el proyecto el vecino próximo siempre será la estación base.

Por otro lado, una vez se ha realizado todo el proceso anterior, los nodos ya cuentan con toda la información necesaria para cifrar un mensaje. Para ello, se utiliza el método *Encrypt()* de la clase *Key_RSA* [45] que necesita de la clave pública del nodo que recibirá el mensaje. Se utiliza como prueba para observar el correcto funcionamiento del algoritmo, una de las medidas que el sensor es capaz de aportar, la luminosidad. Se enviarán mensajes periódicos con los datos de la luminosidad del sensor cifrados a la estación base para que ésta posteriormente los descifre y muestre los resultados. Dichos datos, al igual que el módulo de la clave, también deberán ser enviados en dos paquetes ya que el método *Encrypt()* devuelve un vector de tamaño 128 bytes.

3.6 DIAGRAMA DE SECUENCIA CON RSA

El diagrama de secuencia muestra la interacción entre el sensor que actúa como estación base y el nodo en la versión definitiva del algoritmo.

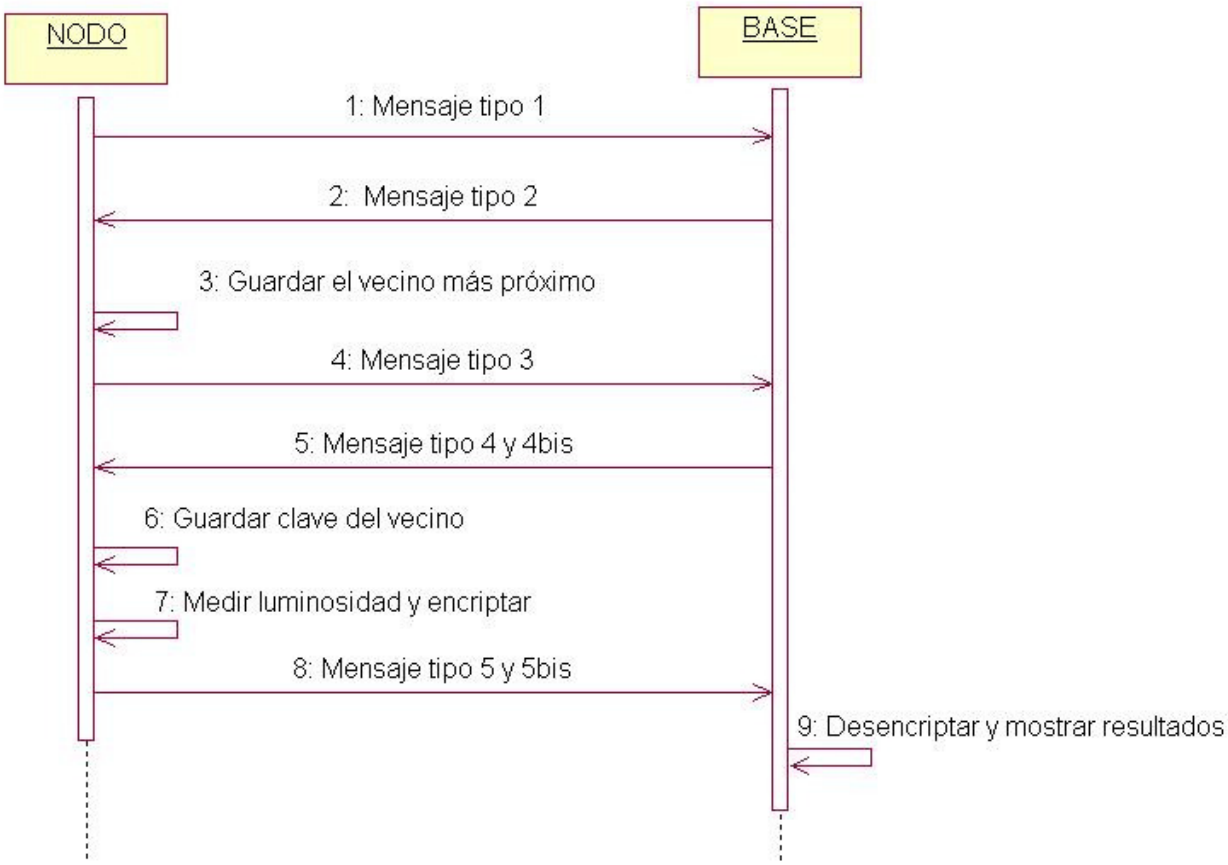


DIAGRAMA 1: Diagrama de secuencia de RSA

Mensaje tipo 1: tipo mensaje, identificador nodo.
Mensaje tipo 2: tipo mensaje, identificador nodo, coste.
Mensaje tipo 3: tipo mensaje, identificador nodo, vecino.
Mensaje tipo 4: tipo mensaje, identificador nodo, primera parte de la clave.
Mensaje tipo 4bis: tipo mensaje, identificador nodo, segunda parte de la clave.
Mensaje tipo 5: tipo mensaje, identificador nodo, primera parte datos encriptados.
Mensaje tipo 5bis: tipo mensaje, identificador nodo, segunda parte datos encriptados.

Es importante resaltar que las primeras instrucciones que cada sensor lleva a cabo cuando comienza la ejecución del programa están destinadas a definir las claves pública y privada de cada uno de ellos. Es un proceso manual, ya que tanto la estación base como cada uno de los nodos de la red deben conocer ambas claves y éstas deben ser diferentes para cada uno de ellos por lo que el programador deberá ir variando manualmente las claves antes de transferir el programa al sensor en caso de tener varios nodos. En el caso concreto que trata el proyecto esto no es necesario ya que tan sólo se

utiliza un nodo y la estación base. De igual manera, la estación base contiene las claves pública y privada de todos los nodos que conformarán la red para luego poder atender las peticiones de los nodos.

Cuando comienza la interacción entre las dos clases, la clase nodo envía ininterrumpidamente un mensaje *broadcast* de tipo uno que contiene su identificador hasta que esta misma clase detecte que ha recibido un paquete.

Por otro lado, la clase base al comienzo de su ejecución se encuentra esperando hasta recibir un paquete de tipo uno. Cuando lo recibe, guarda la dirección desde donde proviene el paquete para saber a dónde deberá enviar el siguiente mensaje que contendrá el identificador del mensaje, dos, su identificador y el coste hasta dicho nodo que en el caso concreto de la red con la que se trabaja va a ser cero ya que sólo contiene dos nodos.

El nodo sale del bucle inicial tras recibir un mensaje y después de verificar que se trata de un mensaje de tipo dos, la clase guarda el identificador, el coste y la dirección del nodo de donde proviene el paquete en una estructura llamada *proxvecino* que previamente se había inicializado con valores por defecto. Cuando concluye este proceso, se envía un mensaje del tipo tres al nodo más cercano guardado en dicha estructura y que en este caso siempre será la estación base, con el identificador del nodo que lo envía y el identificador de su vecino.

En caso de que la red contase con múltiples sensores el proceso anterior se llevaría a cabo en dos partes. En primer lugar se almacenaría en una lista los datos de todos los nodos que mandasen paquetes de tipo dos y al mismo tiempo se guardaría en la estructura *proxvecino* la información del sensor que tuviese coste más pequeño, es decir del que se encontrase más cerca.

La estación base, al recibir un mensaje de tipo tres, crea el mensaje de tipo cuatro y cuatro bis con la clave pública del vecino que aparecía en el paquete recibido. Ya que el exponente público es siempre una constante no es necesario mandar esta información en el paquete sino que sólo es necesario el módulo que será único para cada sensor. El módulo tiene una longitud de 128 bytes por lo que no es posible enviarlo en una vez. Es por esto por lo que se envían dos mensajes seguidos, el cuatro y el cuatro bis.

En el momento en el que el nodo recibe y guarda la clave pública del vecino más cercano ya puede codificar cualquier tipo de datos por lo que utilizará el sensor para medir la luminosidad y tras codificarla utilizando la clave pública del nodo que recibirá el mensaje, enviará varios mensajes (mensaje tipo cinco y cinco bis) con la información de la luminosidad cifrada. Las instrucciones para llevar a cabo este proceso son las siguientes:

- Se mide la luminosidad y se almacena en una variable light de tipo uint:
  light = (ushort)_sensor.Light;

- Hay que convertir el dato de la luminosidad a string y se almacenarlo en lightstring:
  lightstring = light + " ";

- Se debe hacer otra conversión a byte [] para poder encriptarlo ya que el método *Encrypt()* no acepta otro tipo de argumento:
  byte[] lightBytes = Encoding.UTF8.GetBytes(lightstring);

- Se genera una nueva instancia de la clase *KEY_RSA* pasando como argumentos el módulo que se ha recibido en los mensajes anteriores y el exponente público:
  Key_RSA encryptor = new Key_RSA(llave.Modulo, expopublico);

- Se cifran los datos los datos de la luminosidad convertidos en bytes:
  byte [] lightcifrado = encryptor.Encrypt(lightBytes, 0, lightBytes.Length, null);

Tras cifrar la información el nodo manda los datos cifrados mediante dos mensajes de tipo cinco y cinco bis ya que como ocurría con el módulo tiene una longitud de 128 bytes y no es posible mandarlo en un paquete.

El último paso implica que la estación base, en el momento que recibe un mensaje de tipo cinco y cinco bis y recupera la información transmitida por el nodo, la descifrar mediante su clave pública y privada. La estación base muestra por pantalla los datos de la luminosidad obtenidos tras descifrar el mensaje ya que aunque a priori parecía mejor la idea de escribirlos en un fichero de texto para facilitar su posterior estudio, la clase necesaria para este proceso, *StreamWriter* [46], es incompatible con el *Micro Framework* de los sensores. Este proceso está recogido en el código como se muestra a continuación:

- Se genera una nueva instancia de la clase *KEY_RSA* pasando como argumentos el módulo de la estación base y el exponente privado ya que es el que se necesita para desencriptar
  Key_RSA encryptor = new Key_RSA(modulo, expoprivado);

- Se descifran los datos recibidos en el mensaje
  byte[] lightBytes = encryptor.Decrypt(lightcifrado, 0, lightcifrado.Length, null);

En este paso es donde se produce el problema más grande en la implementación del algoritmo ya que como se ha comentado anteriormente *Key_RSA* [45] es la única clase de encriptación con RSA soportada por el *Micro Framework* pero tras varias pruebas en las que saltaba una excepción y buscando información sobre ello se puede concluir que aunque el algoritmo está implementado correctamente y debería funcionar algunos dispositivos como los sensores utilizados [3] no soportan el proceso de descifrado de mensajes. Este es el motivo por el que se ha creado otra versión del algoritmo para poder llevar a cabo alguna medida del tiempo que le cuesta ejecutarse al algoritmo y para justificar que utilizando otro tipo de encriptación el algoritmo funciona correctamente.

3.7 DIAGRAMA DE SECUENCIA CON XTEA

La solución propuesta consiste en el mismo algoritmo que hemos utilizado anteriormente pero introduciendo un método de cifrado de datos diferente, XTEA [47].

3.7.1 TEA y XTEA

Tiny Encryption Algorithm, TEA [48] es un algoritmo de criptografía que destaca por su simplicidad. Fue diseñado por David Wheeler [49] y Roger Needham [50] en 1994. TEA opera en bloques de 64 bits y usa una clave de 128 bits. La primera versión de TEA publicada tenía algunos puntos débiles por lo que se completó con una segunda versión XTEA [47] que incorpora extensiones para hacerlo más seguro. XTEA opera con bloques de tamaño arbitrario en lugar de los bloques de 64 bits del original. XTEA utiliza una sola clave para cifrar y descifrar los datos. Es por esta razón por la que no tiene sentido utilizar este tipo de codificación en el escenario que se está tratando ya que si un nodo debe mandar su clave para que otro cifre la información y sólo tiene una clave, cualquier otro nodo de la red que conozca su clave también podrá descifrar dicha información, lo cual supone la pérdida de la confidencialidad de los datos. De ahí que sólo se utilice este tipo de encriptación como prueba para comprobar que el algoritmo funciona correctamente.

3.7.2 Cambios en el código

El diagrama de secuencia muestra la interacción entre el sensor que actúa como estación base y el nodo en la versión de prueba del algoritmo. Se puede considerar prácticamente igual ya que tan sólo cambia la forma de cifrar y descifrar los datos pero el proceso hasta ese momento es el mismo.

Los mensajes que se intercambian son:

Mensaje tipo 1: tipo mensaje, identificador nodo.
Mensaje tipo 2: tipo mensaje, identificador nodo, coste.
Mensaje tipo 3: tipo mensaje, identificador nodo, vecino.
Mensaje tipo 4: tipo mensaje, identificador nodo, primera parte de la clave.
Mensaje tipo 4bis: tipo mensaje, identificador nodo, segunda parte de la clave.
Mensaje tipo 5: tipo mensaje, identificador nodo, datos cifrados.

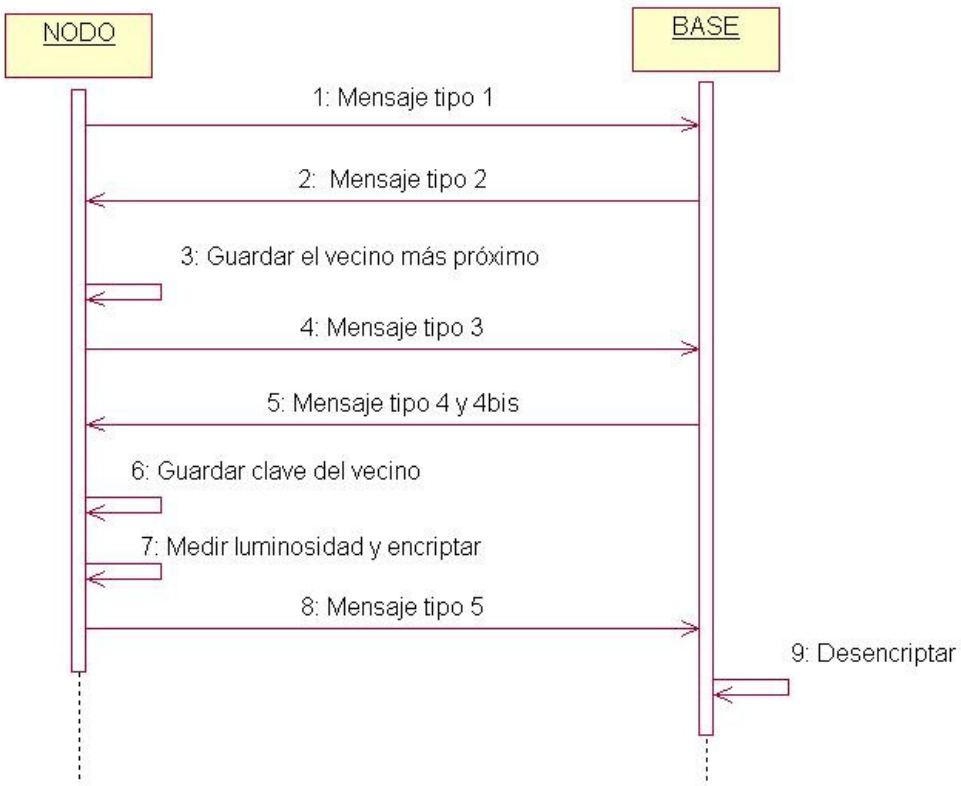


DIAGRAMA 2: Diagrama de secuencia de XTEA

La ejecución del algoritmo se lleva a cabo de manera idéntica que en el caso anterior hasta el momento en el que ya se han enviado las claves y se va a comenzar con el cifrado. Se muestra el código utilizado en este caso:

- La clave utilizada para este algoritmo es:
  byte[] XTEA_key = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };

- Creamos una instancia de la clase Key_TinyEncryptionAlgorithm a la que hay que pasarle como parámetro la clave:
  Key_TinyEncryptionAlgorithm xtea = new Key_TinyEncryptionAlgorithm(XTEA_key);

- Se mide la luminosidad siguiendo el mismo proceso que en el caso anterior y por último se convierte en un array de bytes:
  byte[] lightBytes = UTF8Encoding.UTF8.GetBytes(lightstring);

- Por último se cifra la información de la siguiente manera:
  byte[] lightcifrado = xtea.Encrypt(lightBytes, 0, lightBytes.Length, null);

Tras esto, el nodo envía la información a la estación base mediante un solo mensaje de tipo cinco ya que la longitud de la cadena cifrada en este caso es mucho menor que en caso anterior.

Una vez que la estación base ha recibido los datos cifrados los recupera del paquete para poder descifrarlos. Las instrucciones necesarias para este proceso son las siguientes:

- Creamos una instancia de la clase Key_TinyEncryptionAlgorithm a la que hay que pasarle como parámetro la clave:
  Key_TinyEncryptionAlgorithm xtea = new Key_TinyEncryptionAlgorithm(XTEA_key);

- Se descifra la información de la siguiente manera:
  byte[] decrypted_bytes = xtea.Decrypt(lightcifrado, 0, lightcifrado.Length, null);

De esta manera, se consigue un algoritmo que cifra y descifra datos y así podemos utilizarlo para llevar a cabo alguna medición.

4. RESULTADOS OBTENIDOS

En este capítulo se muestran los resultados obtenidos tras una serie de ejecuciones del algoritmo así como algunas gráficas comparativas entre ellos.

4.1 RESULTADOS OBTENIDOS

Para mostrar los datos obtenidos, lo más sencillo hubiese sido contar con otro sensor del mismo tipo que los usados hasta el momento y cargar el programa *Xsniffer* [51] ya que es un programa que escucha los mensajes de radio y los transmite a través del protocolo USBSPOT al puerto USB. Esta solución no fue viable ya que no se contaba con el material suficiente.

La alternativa pensada consistía en utilizar un sensor del mismo fabricante pero del tipo Iris [3] para cargar en él su correspondiente *sniffer* y así poder tener referencia del tiempo que le costaba ejecutarse al algoritmo. Las primeras pruebas no recogían nada de información por lo que se dedujo que al tratarse de mensajes de tipos diferentes dicho *sniffer* no era capaz de escucharlos. Además, como no se había barajado la posibilidad de tener que utilizar estos sensores, la clase elegida para recibir y mandar mensajes era la clase *radio* [52] con la cual sólo podían intercambiar mensajes las motas Imote2. En vistas de que cambiar todo el código no era la mejor solución se propuso una nueva alternativa que era hacer que nuestro programa mandase un mensaje al comienzo de la ejecución y otro al final de la misma utilizando la clase *TosRadio* [52] para que el sensor Iris pudiese interceptar los mensajes y así medir el tiempo de espera entre un mensaje y otro pero no se consiguió hacerla funcionar.

Es por estas razones por la que se recurrió a la solución menos elegante pero la última posible, medir manualmente el tiempo haciendo que se encendiese una luz al principio de la ejecución y otra de otro color al final. Otra alternativa era ir mostrando por pantalla los mensajes una vez se iban descifrando pero el retardo era mayor. A continuación se muestran los resultados obtenidos en series de doscientas, quinientas y mil interacciones.

4.1.1 Ejecuciones de doscientas interacciones

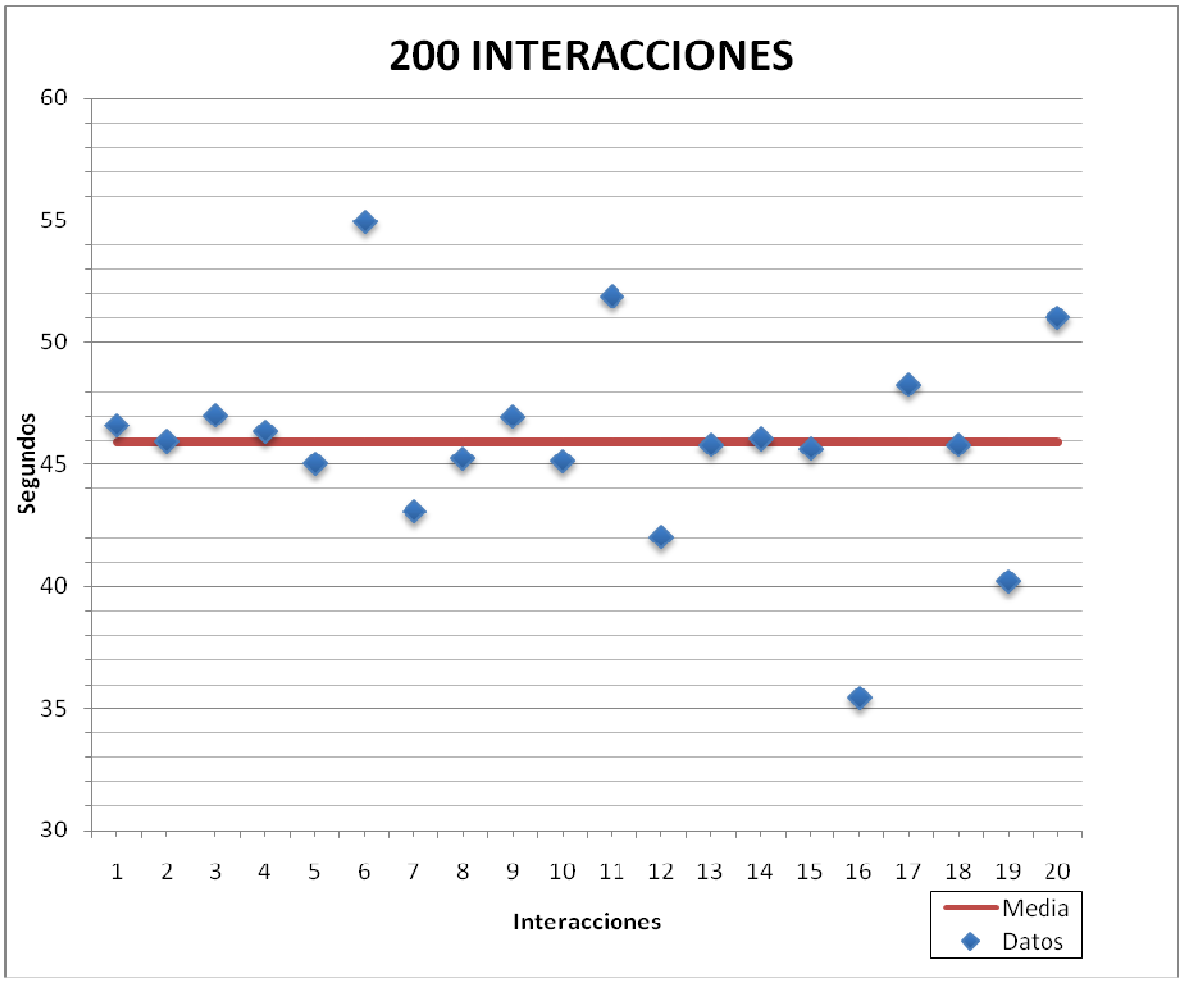
En la tabla se muestran los resultados (en minutos, segundos y centésimas de segundo) de las distintas ejecuciones del algoritmo haciendo doscientas interacciones de la fase de cifrado de los datos, envío de los mismos, almacenado de la información en el otro nodo y su posterior descifrado.

2,46,58	2,54,96	2,51,89	2,35,42
2,45,94	2,43,09	2,42,03	2,48,53
2,47,01	2,45,23	2,45,80	2,45,78
2,46,35	2,46,94	2,46,06	2,40,20
2,45,03	2,45,13	2,45,63	2,51,01

TABLA 2: La tabla muestra los resultados en minutos, segundos y centésimas de segundo obtenidos tras la medida del tiempo de ejecuciones de 200 interacciones.

Estudiando los tiempos medidos y representados en la tabla anterior se puede calcular que al algoritmo le cuesta una media de 2.45.9155 minutos hacer doscientas interacciones y por consiguiente tardará 0.8296 segundos de media en realizar cada interacción. Los datos medidos cuentan con una desviación típica de 3.9975 que en este caso, al tratarse de series a las que le cuesta pocos minutos acabar, podría tomarse como un tiempo relativamente alto, pero teniendo en cuenta que las medidas no se han tomado de la manera más exacta posible se puede concluir que el algoritmo es bastante constante en la ejecución de las interacciones.

A continuación se muestra una gráfica que representa la dispersión de los datos (en azul) así como la media de éstos (rojo) para facilitar su estudio. En el eje Y de la gráfica sólo se han tenido en cuenta los segundos ya que los minutos eran constantes en todas las repeticiones. Es por esto, por lo que puede parecer que la desviación de los datos es muy alta pero tiene un valor de 3.9975 segundos.



GRÁFICA 1: Tiempos para 200 interacciones

4.1.2 Ejecuciones de quinientas interacciones

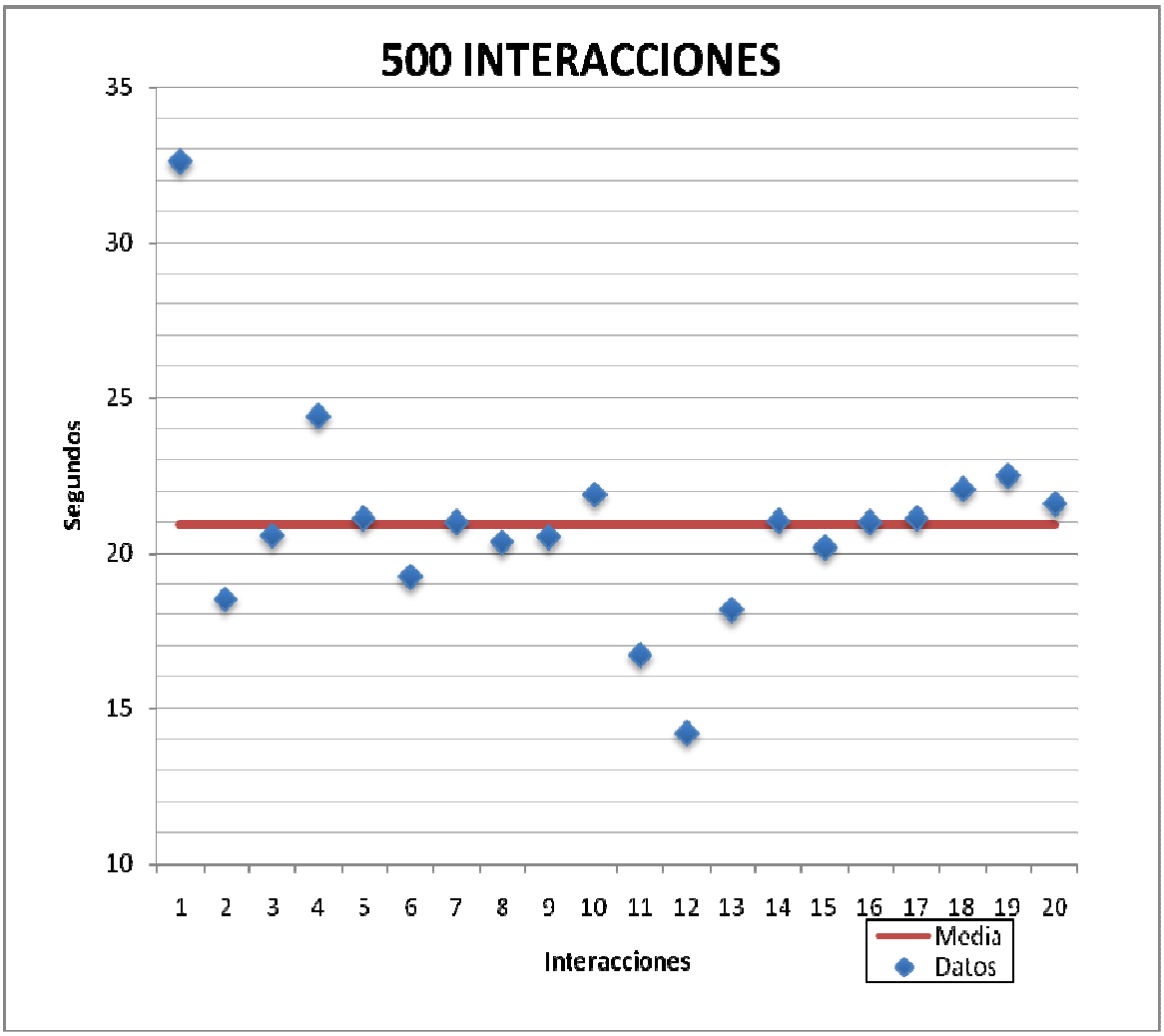
En la tabla se muestran los resultados (en minutos, segundos y centésimas de segundo) de las distintas ejecuciones del algoritmo haciendo quinientas interacciones de la fase de cifrado de los datos, envío de los mismos, almacenado de la información en el otro nodo y su posterior descifrado.

6.32.60	6.19.23	6.16.72	6.20.98
6.18.53	6.20.98	6.14.21	6.21.10
6.20.58	6.20.36	6.18.17	6.22.05
6.24.40	6.20.51	6.21.04	6.22.48
6.21.13	6.21.90	6.20.18	6.21.58

TABLA 3: La tabla muestra los resultados en minutos, segundos y centésimas de segundo obtenidos tras la medida del tiempo de ejecuciones de 500 interacciones.

Estudiando los tiempos medidos y representados en la tabla anterior se puede calcular que al algoritmo le cuesta una media de 6.20.9365 minutos hacer quinientas interacciones y por consiguiente tardará 0.7619 segundos de media en ejecutar cada interacción. Los datos medidos cuentan con una desviación típica de 3.418125 segundos lo cual muestra que el algoritmo es bastante constante teniendo en cuenta que las medidas no se han tomado de la manera más exacta posible y tratándose de unos tiempos que comienzan a ser más elevados, en torno a los seis minutos.

A continuación se muestra una gráfica que representa la dispersión de los datos (en azul) así como la media de éstos (rojo) para facilitar su estudio. A continuación se muestra una gráfica que representa la dispersión de los datos (en azul) así como la media de éstos (rojo) para facilitar su estudio. En el eje Y de la gráfica sólo se han tenido en cuenta los segundos ya que los minutos eran constantes en todas las repeticiones. Es por esto, por lo que puede parecer que la desviación de los datos es muy alta pero tiene un valor de 3.418125 segundos.



GRÁFICA 2: Tiempos para 500 interacciones

4.1.3 Ejecuciones de mil interacciones

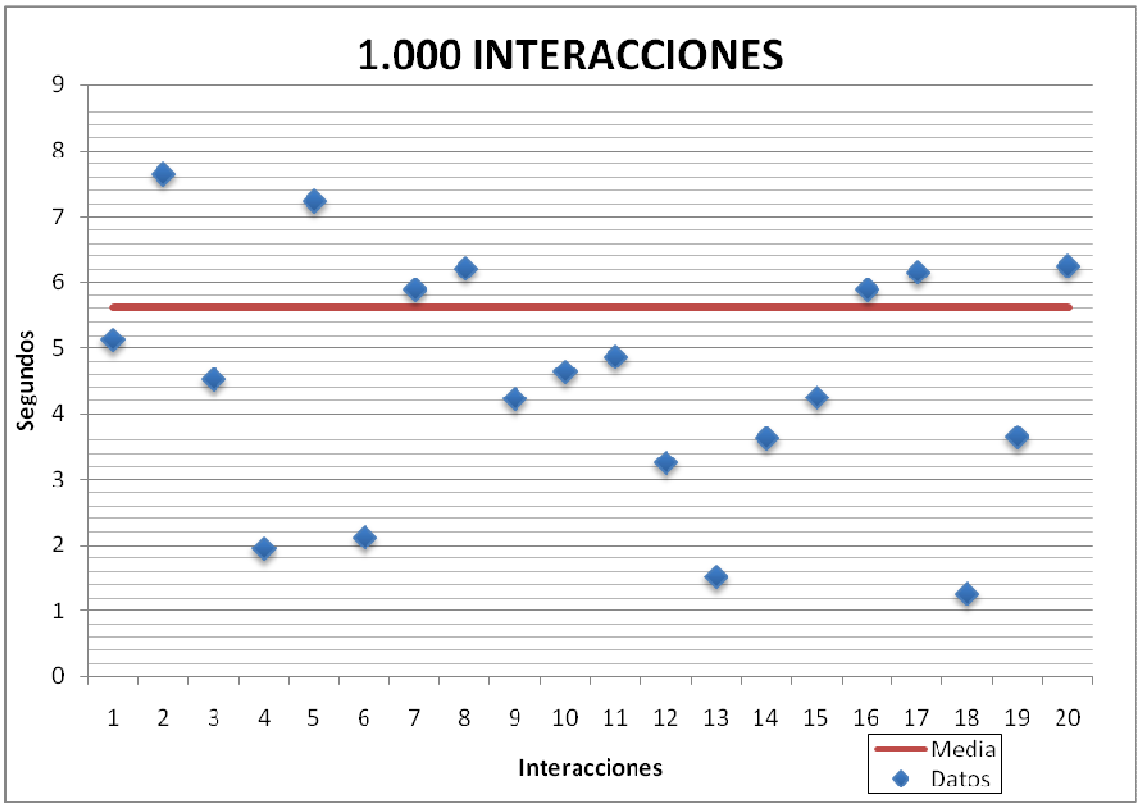
En la tabla se muestran los resultados (en minutos, segundos y centésimas de segundo) de las distintas ejecuciones del algoritmo haciendo mil interacciones de la fase de cifrado de los datos, envío de los mismos, almacenado de la información en el otro nodo y su posterior descifrado.

13.05.12	13.02.12	13.04.86	13.05.88
13.07.65	13.05.88	13.03.76	13.06.14
13.04.52	13.06.21	13.01.52	13.01.25
13.01.94	13.04.23	13.03.63	13.03.65
13.07.23	13.4.64	13.04.25	13.06.25

TABLA 4: La tabla muestra los resultados en minutos, segundos y centésimas de segundo obtenidos tras la medida del tiempo de ejecuciones de 1.000 interacciones.

Estudiando los tiempos medidos y representados en la tabla anterior se puede calcular que al algoritmo le cuesta una media de 13.4.5115 minutos hacer doscientas interacciones y por consiguiente tardará 0.7845 segundos de media en ejecutar cada interacción. Los datos medidos cuentan con una desviación típica de 2.1208 segundos lo cual muestra que el algoritmo es bastante constante teniendo en cuenta que las medidas no se han tomado de la manera más exacta posible y que los datos que se manejan son de trece minutos.

A continuación se muestra una gráfica que representa la dispersión de los datos (en azul) así como la media de éstos (rojo) para facilitar su estudio. A continuación se muestra una gráfica que representa la dispersión de los datos (en azul) así como la media de éstos (rojo) para facilitar su estudio. En el eje Y de la gráfica sólo se han tenido en cuenta los segundos ya que los minutos eran constantes en todas las repeticiones. Es por esto, por lo que puede parecer que la desviación de los datos es muy alta pero tiene un valor de 2.1208 segundos.

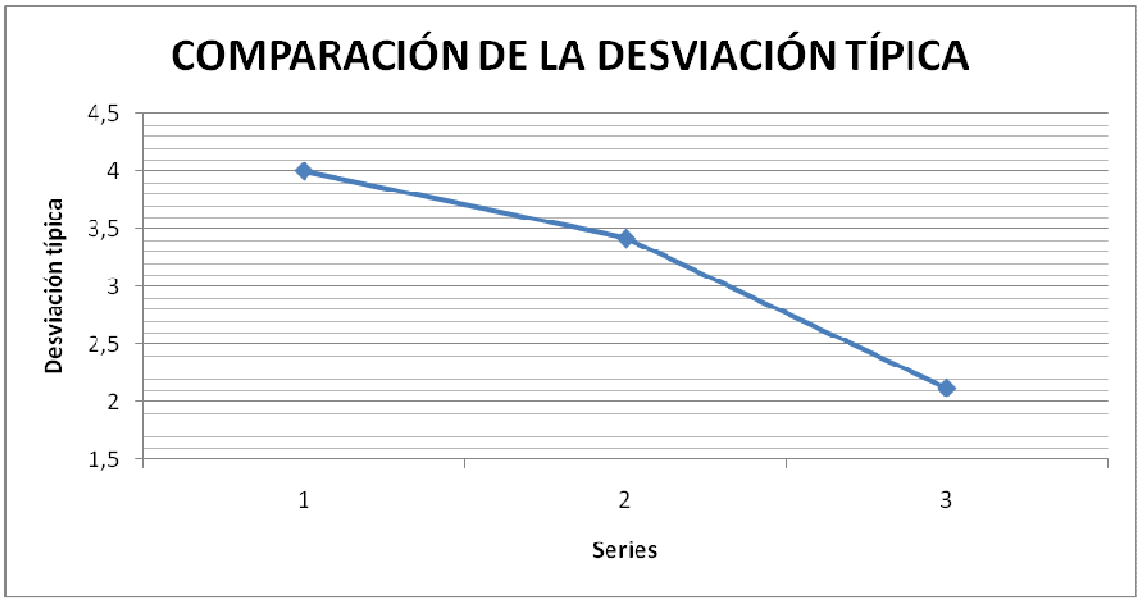


GRÁFICA 3: Tiempos para 1.000 interacciones

4.1.4 Comparativas de los resultados

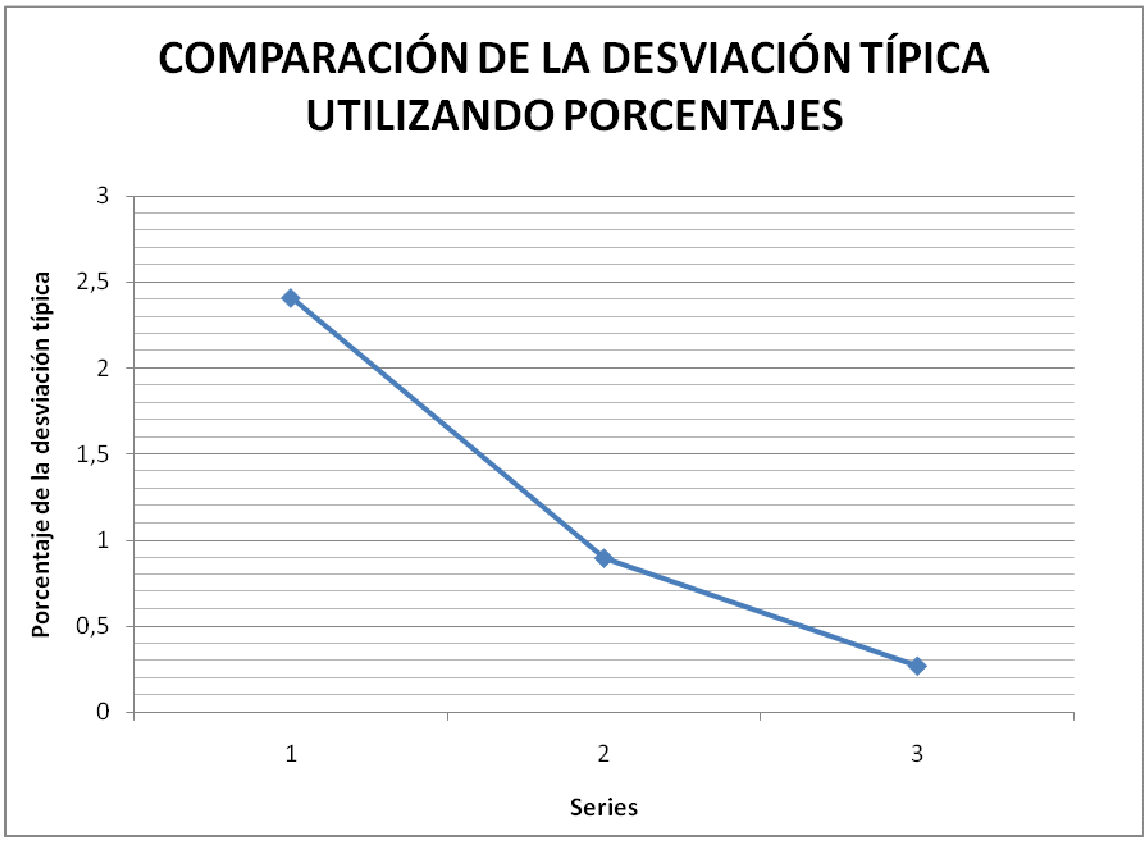
Cabe destacar que en los gráficos anteriores sólo se han representado los segundos ya que los minutos no aportan información para el estudio de la desviación de los datos porque la diferencia de una ejecución a otra con el mismo número de interacciones sólo varía en segundos.

En los dos primeros casos estudiados la desviación típica es muy parecida de lo cual podemos deducir que el algoritmo trabaja de forma constante ya que este margen de error puede estar también asociado a la manera de medir el tiempo. Aun así, cuando se realizan series de quinientas interacciones, el dato de la desviación disminuye ya que al tratarse de tiempos más grandes los segundos de diferencia son menos apreciables. La variación más evidente de la desviación típica se produce en el momento en el que se llevan a cabo muchas interacciones, en este caso mil, ya que al tratarse de tiempos mucho más elevados los márgenes de error son mucho menores. La siguiente gráfica muestra cómo varía la desviación típica en los tres casos en los que se han realizado mediciones.



GRÁFICA 4: Comparativa de la desviación típica

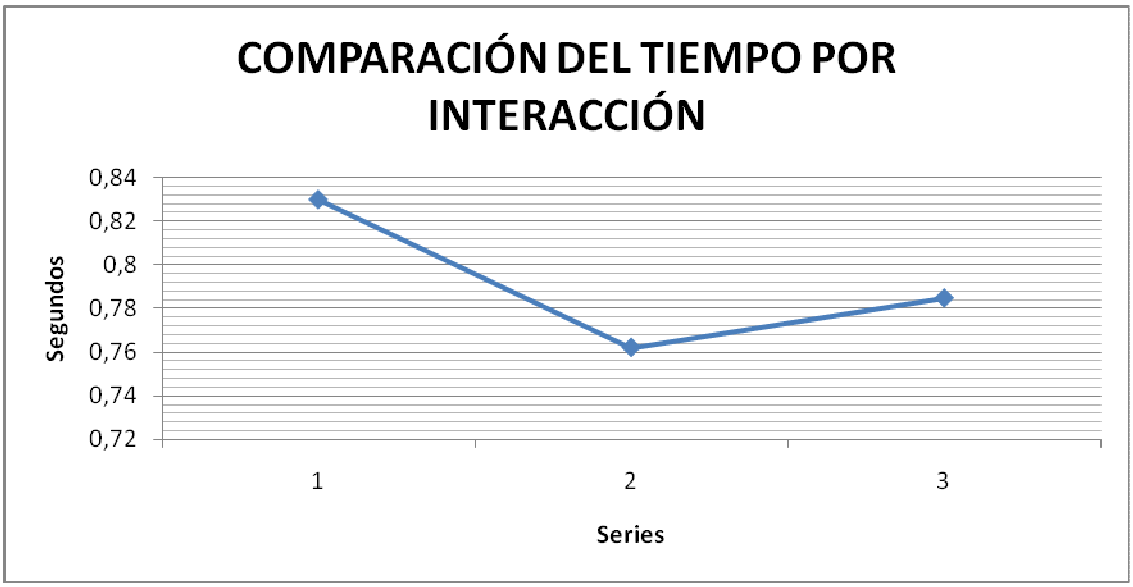
Aunque parece que la desviación típica es muy grande hay que tener en cuenta que en el eje Y los datos se están representando como segundos. Para que la comparación de la desviación típica sea más clara, se ha representado otra gráfica con los valores dados en porcentajes.



GRÁFICA 5: Comparativa de la desviación típica utilizando porcentajes.

La gráfica anterior muestra el porcentaje de desviación típica con respecto a los datos. En la primera serie, que muestra las ejecuciones de doscientas interacciones se puede apreciar que la desviación típica es del 2.4093%, en la segunda serie, quinientas interacciones, es del 0.8972% y en la última, mil interacciones, es de 0.2703%. En esta gráfica, es más fácil apreciar que la desviación típica es pequeña teniendo en cuenta el tipo de medición llevado a cabo y además, se puede observar cómo cuantas más interacciones se hagan y el tiempo vaya aumentando, la desviación se convierte en un dato menor. Este error puede estar relacionado con la forma de medir.

El tiempo medio por interacción varía muy poco en los casos de quinientas y mil interacciones pero en el caso de doscientos éstas diferencias se hacen más visibles. No es algo perceptible ya que se trata de centésimas de segundo y es algo que podríamos relacionar con la forma de medir los datos pero es un dato a tener en cuenta. La siguiente gráfica muestra los valores



GRÁFICA 6: Comparativa del tiempo por interacción

5. CONCLUSIONES Y POSIBLES MEJORAS

En este capítulo se exponen las conclusiones obtenidas tras el desarrollo del proyecto así como posibles mejoras para que el funcionamiento de éste en el futuro reporte más beneficio.

5.1 CONCLUSIONES

El trabajo se ha centrado en crear un mecanismo para el transporte seguro de mercancías para lo cual se ha diseñado un algoritmo capaz de cifrar y descifrar información e intercambiar dicha información entre diferentes sensores. Se ha conseguido que sólo los sensores asociados a la red puedan tener acceso a estos datos gracias a un proceso de intercambio de claves que sólo conoce la estación base y que desvela a los diferentes nodos una vez se han registrado en la red. No se ha podido probar el trabajo con criptografía de claves asimétricas como estaba pensado en un principio porque el sensor no soportaba el proceso de descifrado de los datos pero el algoritmo está implementado correctamente. Si se consiguiese hacer funcionar esta manera de operar se podría garantizar la seguridad en el intercambio de datos. Como prueba de que el algoritmo opera de la forma correcta y aunque no tiene mucho sentido en el escenario ante el que se pensaba trabajar, se ha utilizado otro tipo de criptografía de claves simétricas.

Uno de los inconvenientes a tener en cuenta es que la estación base debe conocer de antemano cuántos nodos formarán la red ya que debe conocer sus claves públicas para más tarde poder enviárselas a los nodos que hagan la petición. En el momento en el que la red vaya a contar con un número muy elevado de nodos sería mejor optar por una base de datos que guarde las claves y de dónde vaya leyendo la estación base.

Tras la finalización del proyecto, cabe destacar que me ha parecido una experiencia muy interesante sobre todo por el tema tratado en él ya que para mí era absolutamente desconocido. A lo largo de la carrera no habíamos estudiado nada relacionado con los sensores lo cual supuso empezar desde cero, pero a la vez me ha ayudado a conocer ciertos aspectos de estos que me parecen muy interesantes dada la gran utilidad de los mismos hoy en día. Todo esto, me ha ayudado a ampliar mis conocimientos en una dirección totalmente distinta, lo cual me parece muy interesante para poder ir definiendo mis preferencias para el futuro.

En algunos momentos ha sido complicado desarrollar el proyecto dada mi inexperiencia en este campo y sobre todo las restricciones de los sensores ya que trabajan con un *API* muy reducida que hace que las alternativas sean muy pocas. Al tratarse de unos sensores relativamente modernos tampoco contaba con mucha información sobre ellos y sobre todo con ejemplos de algoritmos que los utilizasen. El lenguaje de programación aplicado aunque era muy parecido a algunos ya estudiados, también era nuevo para mí.

Con respecto a la criptografía, hasta ahora también desconocida para mí, es un tema muy importante hoy en día ya que cada vez se utilizan más este tipo de plataformas en nuestra sociedad y se debe tener en cuenta la seguridad al utilizarlas. Los datos que manden entre ellas sólo deben ser conocidos por los receptores de dichos datos y para conseguir este objetivo es muy importante la codificación de los mismos. Es un tema en el que me gustaría profundizar más adelante.

En general me parece que los resultados obtenidos son aceptables ya que aunque me he encontrado con muchos problemas, he intentado buscar una alternativa a ellos justificando en todo momento porque he elegido ese camino. Los objetivos marcados y no conseguidos como la utilización de curvas elípticas para la encriptación era algo que

no dependía de mi trabajo personal ya que el *Micro Framework* con el que trabajan los sensores es un poco antiguo y tiene muchas incompatibilidades con funciones más modernas y fáciles de utilizar. Dentro de las opciones con las que contaba debido a los problemas anteriormente comentados creo que los objetivos se han alcanzado en un alto porcentaje ya que se consigue intercambiar claves y cifrar y descifrar datos aunque el tipo de criptografía utilizado para ello no sea el más útil para el escenario a modelar.

Como experiencia personal me parece muy valiosa ya que nunca había desarrollado un proyecto de esta magnitud de manera individual y creo que puede ser muy útil para el futuro laboral.

5.2 MEJORAS

Se podrían incluir una serie de mejoras en el algoritmo lo que reportaría un mayor beneficio con su uso.

La mejora más destacable sería poder probar el algoritmo en una red de sensores más grande para lo que habría que hacer algunos cambios en el código. Se presentan dos soluciones para este caso. Si queremos enviar un mensaje de un nodo a otro, el primero escribe la información encriptada con la clave de su vecino más próximo y se lo manda. Éste lo recibe, cifra la información y repite el proceso anterior mandándolo esta vez a su nodo más cercano. Se repite esta actividad hasta que el mensaje llegue a su destino. Tomando esta alternativa cada nodo sólo tiene que conocer la clave pública de su vecino más cercano lo que facilita el proceso de envío de claves ya que sería similar que en el código desarrollado pero la ejecución consume muchos recursos cifrando y descifrando la información en cada paso de manera innecesaria.

Por otro lado, se puede hacer que los nodos no sólo conozcan la clave de su vecino más próximo sino de todos los nodos de la red. En este caso el nodo cifra la información con la clave pública del nodo al que va dirigido el mensaje, el resto de nodos sólo reenvían el paquete a su vecino más cercano y el destinatario del paquete es el único que descifra la información recibida. Con esta solución, en la que los nodos tienen que conocer las claves públicas de todos, una vez que el resto de nodos vayan contestando a su mensaje de *broadcast* el nodo deberá almacenar todos los identificadores en una lista y guardar también la dirección del vecino más cercano para realizar la petición a la estación base con todos los vecinos que hayan contestado al mensaje.

En cualquiera de los dos casos los mensajes que no están cifrados y que sirven para registrarse en la red y para conseguir las claves del resto de nodos se irían mandando al nodo más cercano repitiendo el proceso hasta que llegasen al nodo destino.

Otro cambio que mejoraría el funcionamiento del algoritmo es tratar de buscar una solución que permita a los sensores estar escuchando y mandando mensajes al mismo tiempo ya que el intento de solucionar este problema con hilos no ha dado buenos resultados. Con esto nos evitaríamos problemas de colisiones de mensajes y se haría el proceso más eficaz.

También sería importante conseguir otro sensor para poder escuchar los mensajes enviados en la red y así poder medir de manera más precisa el tiempo o conseguir que la idea de utilizar otro tipo de sensores sea viable y funcione.

Por otro lado, se podría tratar de buscar alguna solución para transferir los resultados a un fichero en vez de mostrarlos por pantalla ya que, además de ser más elegante, facilita el estudio posterior de dichos datos.

La mejora más importante sería intentar cambiar el *Micro Framework* que utilizan las motas para poder incluir otras clases de criptografía RSA que podrían funcionar.

6. REFERENCIAS

REFERENCIAS

[1] Ecma International , http://www.ecma-international.org/.

[2] ISO (*International Organization for Standardization*), Organización Internacional para la Estandarización. http://www.iso.org/iso/home.html.

[3] Crossbow, http://www.xbow.com/.

[4] S. Madden et al., TAG: A Tiny AGgregation Service for Ad Hoc Sensor Networks, OSDI, Boston, MA, 2002.

[5] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, in: IEEE/ACM Transactions on Networking, vol. 11, 2003, pp. 2–16.

[6] B. Zhou et al., A Hierarchical Scheme for Data Aggregation in Sensor Network, IEEE ICON 04, Singapore, 2004.

[7] S. Lindsey, C. Raghavendra, K.M. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, IEEE Trans. Parallel Distrib. Sys. 13 (9) (2002) 924–935.

[8] M. Ding, X. Cheng, G. Xue, Aggregation tree construction in sensor networks, in: Proceedings of the 58th IEEE Vehicular Technology Conference, vol. 4, 2003, pp. 2168–2172.

[9] G. Di Bacco, T. Melodia, F. Cuomo, A MAC Protocol for Delay-Bounded Applications in Wireless Sensor Networks, Med-Hoc-Net, Bodrum, Turkey, 2004.

[10] W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Trans. Wireless Commun. 1 (4) (2002) 660–670.

[11] O. Younis, S. Fahmy, HEED: a hybrid, energy-efficient distributed clustering approach for ad hoc sensor networks, IEEE Trans. Mobile Comput. 3 (4) (2004) 366–379.

[12] Y. Yao, J. Gehrke, The Cougar approach to in-network query processing in sensor networks, ACM SIGMOD Rec. 31 (3) (2002) 9–18.

[13] S. Chatterjea, P. Havinga, A dynamic data aggregation scheme for wireless sensor networks, in: Proceedings of the Program for Research on Integrated Systems and Circuits, Veldhoven, The Netherlands, 2003.

[14] L. Hu, D. Evans, Secure aggregation for wireless networks, in: Proceedings of the Workshop on Security and Assurance in Ad Hoc Networks, Orlando, FL, 28 January 2003.

[15] B. Przydatek, D. Song, A. Perrig, SIA : secure information aggregation in sensor networks, in: Proceedings of SenSys'03, 2003, pp. 255– 265.

[16] A. Mahimkar, T.S. Rappaport, SecureDAV: a secure data aggregation and verification protocol for wireless sensor networks, in: Proceedings of the 47th IEEE Global Telecommunications Conference (Globecom), November 29–December 3, Dallas, TX, 2004.

[17] H. Çam, S. Ozdemir, P. Nair, D. Muthuavinashiappan, H.O. Sanli, Energy-efficient and secure pattern based data aggregation for wireless sensor networks, Comput. Commun., Elsevier 29 (4) (2006) 446–455.

[18] W. Du, J. Deng, Y.S. Han, P.K. Varshney, A witness-based approach for data fusion assurance in wireless sensor networks, in: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '03), 2003, pp. 1435–1439.

[19] K. Wu, D. Dreef, B. Sun, Y. Xiao, Secure data aggregation without persistent cryptographic operations in wireless sensor networks, Ad Hoc Networks 5 (1) (2007) 100–111.

[20] H.O. Sanli, S. Ozdemir, H. Çam, SRDA: secure reference-based data aggregation protocol for wireless sensor networks, in: Proceedings of the IEEE VTC Fall Conference, Los Angeles, CA, 26–29 September 2004, pp. 4650–4654.

[21] Y. Yang, X. Wang, S. Zhu, G. Cao, SDAP: a secure hop-by-hop data aggregation protocol for sensor networks, in: Proceedings of the ACM MOBIHOC'06, 2006.

[22] S. Ozdemir, Secure and reliable data aggregation for wireless sensor networks, in: H. Ichikawa et al. (Eds.), LNCS 4836, 2007, pp. 102– 109.

[23] D. Westhoff, J. Girao, M. Acharya, Concealed data aggregation for reverse multicast traffic in sensor networks: encryption key distribution and routing adaptation, IEEE Trans. Mobile Comput. 5 (10) (2006) 1417–1431.

[24] C. Castelluccia, E. Mykletun, G. Tsudik, Efficient aggregation of encrypted data in wireless sensor networks, in: Proceedings of the Conference on Mobile and Ubiquitous Systems: Networking and Services, 2005, pp. 109–117.

[25] S. Ozdemir, Secure data aggregation in wireless sensor networks via homomorphic encryption, Journal of The Faculty of Engineering and Architecture of Gazi University 23 (2) (2008) 365–373. ISSN:1304-4915.

[26] Rodhea, C. Rohner, n-LDA: n-layers data aggregation in sensor networks, in: Proceedings of the 28th International Conference on Distributed Computing Systems Workshops, 2008, pp. 400–405.

[27] S. Ozdemir, Concealed data aggregation in heterogeneous sensor networks using privacy homomorphism, in: Proceedings of the ICPS'07: IEEE International Conference on Pervasive Services, Istanbul, Turkey, 2007, pp. 165–168.

[28] W. Zhang, Y. Liu, S.K. Das, P. De, Secure data aggregation in wireless sensor networks: a watermark based authentication supportive approach, Elsevier Pervasive Mobile Comput. 4 (2008) 658–680.

[29] J. Domingo-Ferrer, A provably secure additive and multiplicative privacy homomorphism, in: Proceedings of the Information Security Conference, 2002, pp. 471–483.

[30] T. Okamoto, S. Uchiyama, A New Public-Key Cryptosystem as Secure as Factoring, Advances in Cryptology – EUROCRYPT'98, 1998, pp. 208–318.

[31] A. Josang, R. Ismail, The beta reputation system, in: Proceedings of the 15th Bled Conference Electronic Commerce, 2002.

[32] S. Ganeriwal, M.B. Srivastava, Reputation-based framework for high integrity sensor networks, in: Proceedings of the Second ACM Workshop on Security of Ad Hoc and Sensor Networks, Washington DC, 2004, pp. 66–77.

[33] B. Sun, X. Jin, K. Wu, Y. Xiao, Integration of secure in-network aggregation and system monitoring for wireless sensor networks, in: Proceedings of IEEE International Conference on Communications (IEEE ICC'07), 2007, pp. 1466–1471.

[34] B. Sun, N. Chand, K. Wu, Y. Xiao, Change-point monitoring for secure in-network aggregation in wireless sensor networks, in: Proceedings of IEEE Global Telecommunications Conference, IEEE GLOBECOM, 2007, pp. 936–940.

[35] H. Çam, S. Ozdemir, False data detection and secure data aggregation in wireless sensor networks, in: Yang Xiao (Ed.), Security in Distributed Grid Mobile and Pervasive Computing, Auerbach Publications, CRC Press, 2007.

[36] B. Parekh, H. Çam, Minimizing false alarms on intrusion detection for wireless sensor networks in realistic environments, in: Proceedings of IEEE Military Communications Conference, 2007, pp. 1–7.

[37] S. Ozdemir, Functional reputation based reliable data aggregation and transmission for wireless sensor networks, Elsevier Comput. Commun. 31 (17) (2005) 3941–3953.

[38] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, Impact of network density on data aggregation in wireless sensor networks, in: Proceedings of the 22nd International Conference on Distributed Computing Systems, 2002, pp. 457–458.

[39] ECC, criptografía de curva elíptica: http://www.ldc.usb.ve/~figueira/Cursos/Seguridad/Expo/CurvasElipticas_articulo.pdf

[40] RSA: http://es.wikipedia.org/wiki/RSA

[41] Ron Rivest: http://es.wikipedia.org/wiki/Ron_Rivest

[42] Adi Shamir: http://es.wikipedia.org/wiki/Adi_Shamir

[43] Len Adleman: http://es.wikipedia.org/wiki/Len_Adleman

[44] http://msdn.microsoft.com/es-es/library/system.security.cryptography.rsacryptoserviceprovider_members%28VS.80%29.aspx

[45] http://books.google.es/books?id=TqtZlDCM2QoC&pg=PA216&lpg=PA216&dq=.net+micro+framework++rsa&source=bl&ots=AhLyfPLvN1&sig=VKtxFsTkjFOt7myEpjFu-y5bOIo&hl=es&ei=aJxzTI2OFMb-ObP42aYI&sa=X&oi=book_result&ct=result&resnum=1&ved=0CB8Q6AEwAA#v=onepage&q=.net%20micro%20framework%20%20rsa&f=false

[46] http://msdn.microsoft.com/es-es/library/system.io.streamwriter%28VS.80%29.aspx

[47] http://translate.google.es/translate?hl=es&sl=en&u=http://en.wikipedia.org/wiki/XTEA&ei=aoGGTKGaDJCTjAex16ybCQ&sa=X&oi=translate&ct=result&resnum=1&ved=0CB0Q7gEwAA&prev=/search%3Fq%3Dxtea%26hl%3Des

[48] http://translate.google.es/translate?hl=es&sl=en&u=http://en.wikipedia.org/wiki/XTEA&ei=aoGGTKGaDJCTjAex16ybCQ&sa=X&oi=translate&ct=result&resnum=1&ved=0CB0Q7gEwAA&prev=/search%3Fq%3Dxtea%26hl%3Des

[49] http://translate.google.es/translate?hl=es&sl=en&u=http://en.wikipedia.org/wiki/XTEA&ei=aoGGTKGaDJCTjAex16ybCQ&sa=X&oi=translate&ct=result&resnum=1&ved=0CB0Q7gEwAA&prev=/search%3Fq%3Dxtea%26hl%3Des

[50] http://translate.google.es/translate?hl=es&sl=en&u=http://en.wikipedia.org/wiki/XTEA&ei=aoGGTKGaDJCTjAex16ybCQ&sa=X&oi=translate&ct=result&resnum=1&ved=0CB0Q7gEwAA&prev=/search%3Fq%3Dxtea%26hl%3Des

[51] http://www.google.es/url?sa=t&source=web&cd=1&ved=0CBUQFjAA&url=http%3A%2F%2Fwww.memsic.com%2Fsupport%2Fdocumentation%2Fwireless-sensor-networks%2Fcategory%2F6-user-manuals.html%3Fdownload%3D57%253Aimote2.builder-sdk-manual&rct=j&q=xsniffer%20imote2&ei=Up2ITMnWHcWOjAflxuVn&usg=AFQjCNHusHAW923G4sVHk3sKEf111VNthA&cad=rja

[52] file:///C:/Archivos%20de%20programa/Crossbow/Imote2.Builder/doc/html/namespaces.html

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

"ESTUDIO DE MECANISMOS DE SEGURIDAD EN REDES INALÁMBRICAS DE SENSORES"

Alumno: Patricia Chivite Fernández

Tutor: José Javier Astrain Escola

Pamplona, 15 de Septiembre de 2010

RED INALÁMBRICA

- El término red inalámbrica se utiliza para designar la conexión de nodos sin necesidad de una conexión física (cables). Se da por medio de ondas electromagnéticas.

- Principal ventaja: coste, ya que se elimina todo el cableado y las conexiones físicas entre nodos.

- Desventaja: para este tipo de red se debe tener una seguridad mucho más exigente para evitar intrusiones y escuchas no autorizadas.

DISEÑO DEL PROYECTO

ANÁLISIS DEL PROYECTO

- El objetivo es conseguir un algoritmo para la comunicación segura en una red de sensores inalámbricos .
- ECC mínimas operaciones de cómputo y claves 160 bits.
- RSA (Rivest, Shamir y Adleman), seguridad similar a ECC, mayor consumo recursos, claves 1024 bits.

- El principal uso del algoritmo es para el transporte inteligente de mercancías.
- Trenes, camiones, barcos o aviones.
- Contenedores de diferentes compañías coexistirán.
- Se usan sensores Crossbow: Imote2
- Plataforma .NET
- Lenguaje de programación C#

PROBLEMAS EN LA IMPLEMENTACIÓN DEL ALGORITMO

- Utilizar hilos → duplicar radio.
- Generador de claves: Método ToXmlString() de la clase RSACryptoServiceProvider es el más utilizado para generar aleatoriamente.
  - False → Clave pública
  - True → Clave pública y privada (página 36 memoria)
- Desechar programa generador de claves
- Imposible descifrar información utilizando clase RSA compatible.

GENERAR LAS CLAVES

- RSACryptoServiceProvider vs MetaDataProcessor.exe

- Para crear un par de claves pública y privada:
  - cd "C:\ Archivos de programa\Microsoft .NET Micro Framework\v2.0.3036\Tools"
  - metadataprocessor –create_key_pair c:\private.bin c:\public.bin
  - metadataprocessor –dump_key c:\private.bin >> c:\private.txt
  - metadataprocessor –dump_key c:\public.bin >> c:\public.txt

  - Ejemplo clave privada página 38 memoria

BASE

- Conoce todas las claves y tras un periodo de registro las envía.
- Descifra la información y la muestra

NODO

- Mensajes de registro con la estación base para conseguir la clave del vecino.
- Cifra la información

DIAGRAMA DE SECUENCIA CON RSA

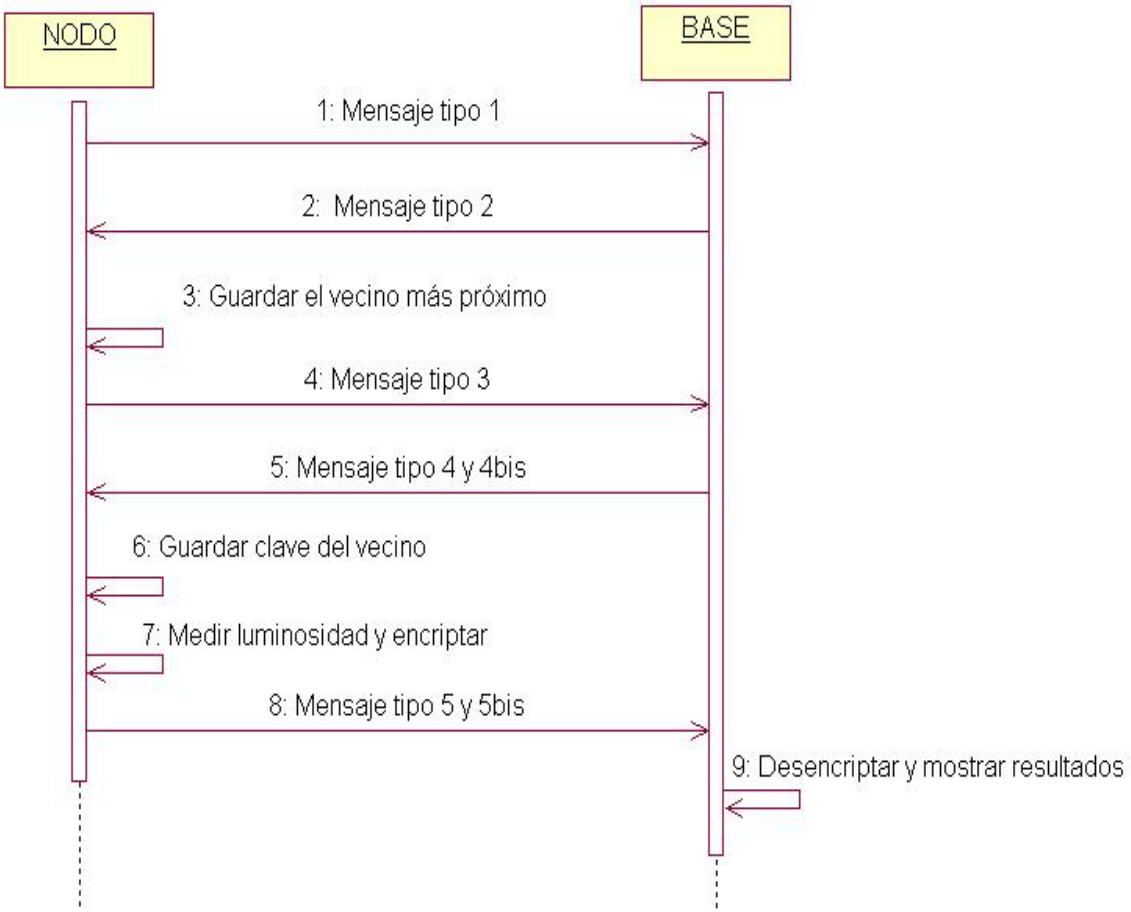


DIAGRAMA 1: Diagrama de secuencia de RSA

- Se debe hacer conversión a byte [] para poder cifrarlo ya que el método *Encrypt()* no acepta otro tipo de argumento:
  - byte[] lightBytes = Encoding.UTF8.GetBytes(lightstring);

- Se genera una nueva instancia de la clase *KEY_RSA* pasando como argumentos el módulo que se ha recibido en los mensajes anteriores y el exponente público:
  - Key_RSA encryptor = new Key_RSA(llave.Modulo, expopublico);

- Se cifran los datos los datos de la luminosidad convertidos en bytes:
  - byte [] lightcifrado = encryptor.Encrypt(lightBytes, 0, lightBytes.Length, null);

- Se genera una nueva instancia de la clase *KEY_RSA* pasando como argumentos el módulo de la estación base y el exponente privado ya que es el que se necesita para descifrarlo
  - Key_RSA encryptor = new Key_RSA(modulo, expoprivado);

- Se descifran los datos recibidos en el mensaje
  - byte[] lightBytes = encryptor.Decrypt(lightcifrado, 0, lightcifrado.Length, null);

- Problema: los sensores no soportan el proceso de descifrado de datos.

- Posible alternativa: utilizar otro algoritmo de criptografía para llevar a cabo alguna medida.

DIAGRAMA DE SECUENCIA CON XTEA

- TEA (Wheeler y Needham 1994) destaca por su simplicidad. Usa una clave de 128 bits. Debido a sus puntos débiles se completó con una segunda versión XTEA que es la que utiliza.

- Una sola clave

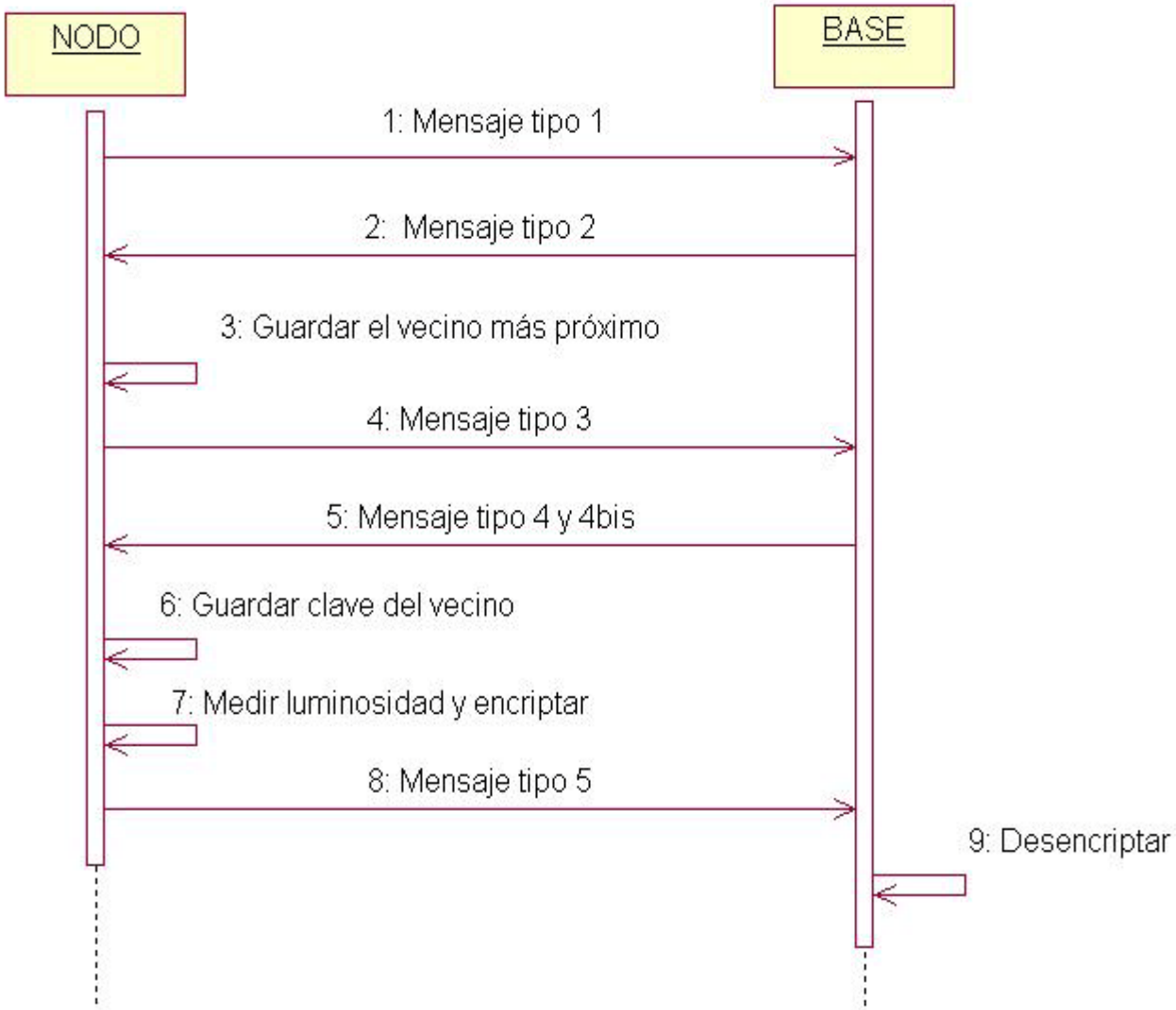


DIAGRAMA 2: Diagrama de secuencia de XTEA

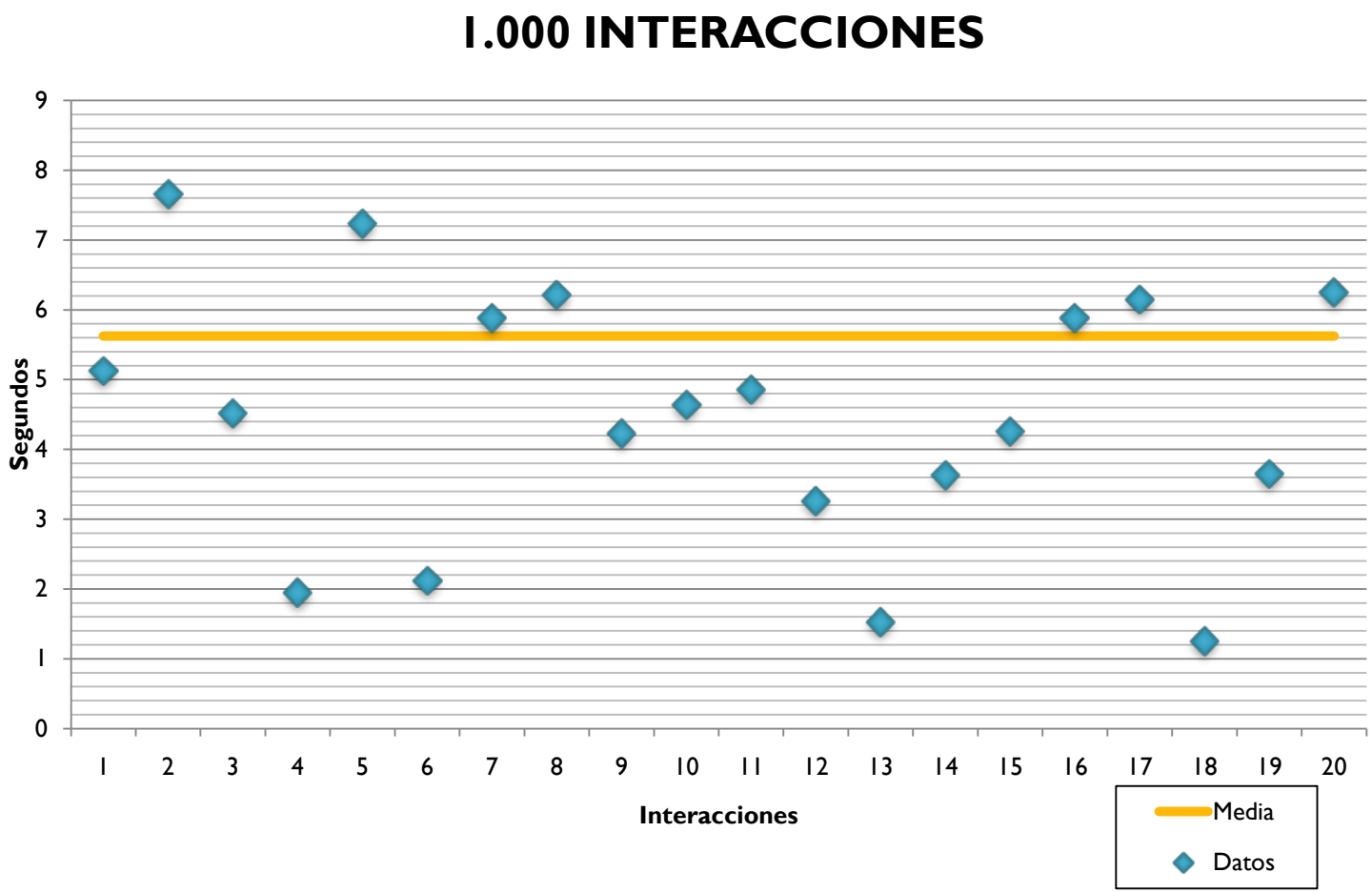
- Creamos una instancia de la clase Key_TinyEncryptionAlgorithm a la que hay que pasarle como parámetro la clave:
  - Key_TinyEncryptionAlgorithm xtea = new Key_TinyEncryptionAlgorithm(XTEA_key);
- Se mide la luminosidad siguiendo el mismo proceso que en el caso anterior y por último se convierte en un array de bytes:
  - byte[] lightBytes = UTF8Encoding.UTF8.GetBytes(lightstring);
- Por último se cifra la información de la siguiente manera:
  - byte[] lightcifrado = xtea.Encrypt(lightBytes, 0, lightBytes.Length, null);

- Creamos una instancia de la clase Key_TinyEncryptionAlgorithm a la que hay que pasarle como parámetro la clave:
  - Key_TinyEncryptionAlgorithm xtea = new Key_TinyEncryptionAlgorithm(XTEA_key);

- Se descifra la información de la siguiente manera:
  - byte[] decrypted_bytes = xtea.Decrypt(lightcifrado, 0, lightcifrado.Length, null);

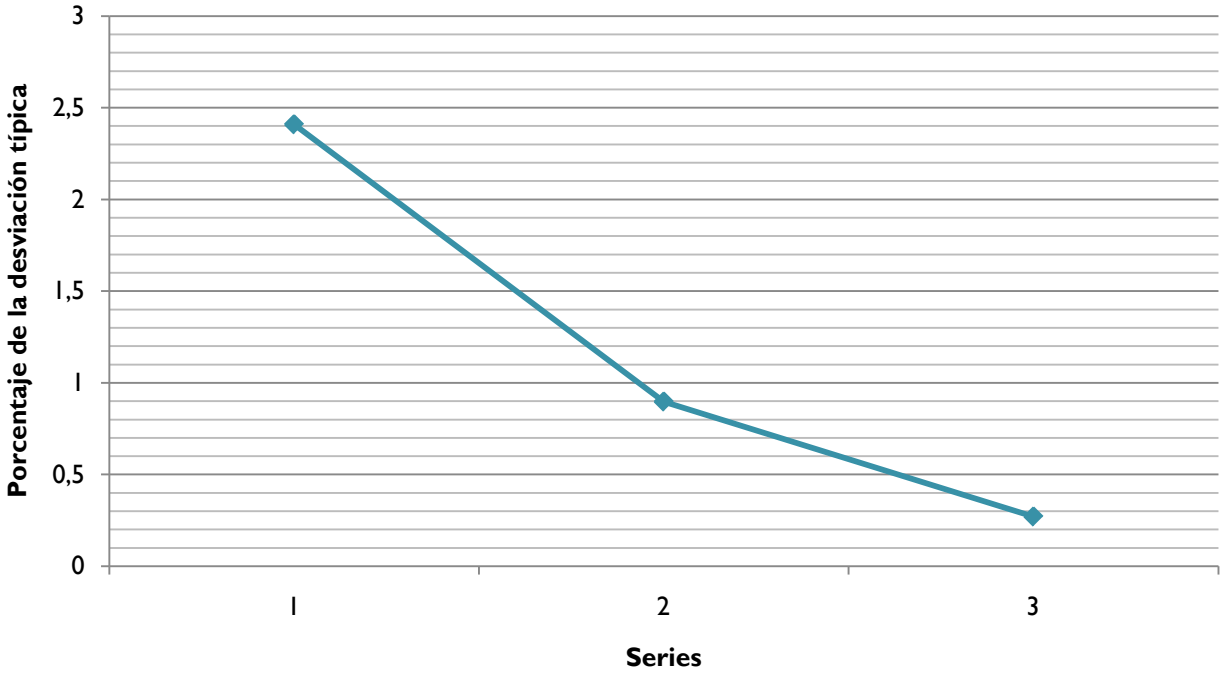
RESULTADOS OBTENIDOS

- Para medir:
- Otro sensor y Xsniffer()
- Un sensor Iris
- Manual
  - 200 interacciones (2.45.9155 minutos)
  - 500 interacciones (6.20.9365 minutos)
  - 1000 interacciones (13.04.5115 minutos)
  - Comparativas



GRÁFICA: Tiempos para 1.000 interacciones

COMPARACIÓN DE LA DESVIACIÓN TÍPICA UTILIZANDO PORCENTAJES



GRÁFICA: Comparativa de la desviación típica utilizando porcentajes.

200 interacciones → 2.4093 %
500 interacciones → 0.8972 %
1.000 interacciones → 0.2703 %

XSNIFFER()

ElapsedTime	Addr	RF	Type	Grp	Len	1	2	3	4	5	6	7	8	9
0:00:11.187	Bcast		255	255	92	255	255	1	0	1	0	0	0	0
0:00:11.187	Bcast		255	255	92	255	255	1	0	1	0	0	0	0
0:00:11.187	Bcast		255	255	92	255	255	2	0	0	0	0	0	0
0:00:11.187	Bcast		255	255	92	255	255	3	0	1	0	0	0	0
0:00:11.187	Bcast		255	255	92	255	255	4	0	0	0	215	199	202
0:00:13.156	Bcast		255	255	92	255	255	41	0	0	0	85	136	224
0:00:15.171	Bcast		255	255	92	255	255	5	0	1	0	21	0	3
0:13:16.468	Bcast		255	255	92	255	255	5	0	1	0	21	0	75

Prueba con Xsniffer para 1.000 interacciones

CONCLUSIONES Y POSIBLES MEJORAS

CONCLUSIONES

- Objetivo cumplido (claves simétricas, asimétricas)
- Sensores y criptografía temas desconocidos
- API reducida
- Experiencia personal

MEJORAS

- Probar el algoritmo en una red de sensores más grande.
- Mandar y recibir mensajes al mismo tiempo (hilos)
- Utilizar otro sensor para medir resultados
- Cambiar *Micro Framework*